(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 807 181**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **00 07244**

(51) Int Cl[7] : **G 06 F 19/00**

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 06.06.00.

(30) **Priorité :** 03.04.00 FR 00004207.

(43) **Date de mise à la disposition du public de la demande :** 05.10.01 Bulletin 01/40.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Ce dernier n'a pas été établi à la date de publication de la demande.*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *MAIM ENRICO* — FR.

(72) **Inventeur(s) :** MAIM ENRICO.

(73) **Titulaire(s) :**

(74) **Mandataire(s) :**

(54) PROCEDE ET SYSTEME DE GESTION, DANS UN POSTE INFORMATIQUE, DE L'AFFICHAGE DE FENETRES LORS D'UNE OPERATION DE GLISSER-DEPOSER.

(57) L'invention propose un procédé de gestion de l'affichage, sur l'écran d'affichage d'un poste informatique, de fenêtres d'un système d'exploitation lors d'une opération de glisser-déposer d'un objet contenu dans une première fenêtre (F1) vers une seconde fenêtre (F2), les première et seconde fenêtres étant affichées de telle sorte que la première fenêtre recouvre partiellement la seconde fenêtre.

ce procédé est remarquable en ce qu'il comprend les étapes successives suivantes:

(a) détection des mouvements d'un pointeur actionné par l'utilisateur au cours d'une opération de glisser d'un objet préalablement saisi dans la première fenêtre, pour déterminer si ledit pointeur se situe dans la zone de la seconde fenêtre non recouverte par la première fenêtre, et

(b) lorsque ledit pointeur a séjourné dans la zone de la seconde fenêtre non recouverte par la première fenêtre et qu'au moins une autre condition est remplie, affichage de la seconde fenêtre (F2) dans son entier de manière à rendre accessible au pointeur l'ensemble de ladite seconde fenêtre.

FR 2 807 181 - A1

La présente invention concerne d'une façon générale la gestions de l'affichage des fenêtres dans un système d'exploitation à fenêtres d'un ordinateur.

On connaît, notamment dans les systèmes d'exploitation classiques tels que « Windows » et « MacOS » (Marques déposées), un mécanisme de gestion de fenêtres pouvant se recouvrir entre elles partiellement ou totalement, mais dans lequel les opérations de déplacement ou de « glisser-déposer » d'un objet d'une fenêtre vers une autre fenêtre (typiquement à l'aide d'un pointeur et des boutons d'un dispositif d'entrée de type souris) peuvent être rendus délicats dans le cas où la fenêtre d'origine recouvre une partie de la fenêtre de destination. En particulier, pour être certain de pouvoir atteindre une région quelconque de la fenêtre de destination lors de cette opération de déplacement ou de glisser-déposer, l'utilisateur doit préalablement déplacer et/ou redimensionner à l'aide du pointeur de la souris soit la fenêtre d'origine, soit la fenêtre de destination, de manière à ce que la première ne recouvre plus la deuxième. On comprend que ceci est fastidieux.

La présente invention vise à proposer un procédé de gestion de l'affichage de fenêtres d'un système d'exploitation qui, tout en étant simple à mettre en œuvre au niveau du système d'exploitation, facilite grandement les opérations de déplacement d'objets (fichiers, applications, dossiers, etc.) entre deux fenêtres qui se recouvrent partiellement, et en particulier ne nécessite aucune modification de l'agencement (déplacement, redimensionnement, etc.) desdites fenêtres.

Ainsi l'invention propose un procédé de gestion de l'affichage, sur l'écran d'affichage d'un poste informatique, de fenêtres d'un système d'exploitation lors d'une opération de glisser-déposer d'un objet contenu dans une première fenêtre (F1) vers une seconde fenêtre (F2), les première et seconde fenêtres étant affichées de telle sorte que la première fenêtre recouvre partiellement la seconde fenêtre, caractérisé en ce qu'il comprend les étapes successives suivantes :

(a) détection des mouvements d'un pointeur actionné par l'utilisateur au cours d'une opération de glisser d'un objet préalablement saisi dans la première fenêtre,

pour déterminer si ledit pointeur se situe dans la zone de la seconde fenêtre non recouverte par la première fenêtre, et

(b) lorsque ledit pointeur a séjourné dans la zone de la seconde fenêtre non recouverte par la première fenêtre et qu'au moins une autre condition est remplie, affichage de la seconde fenêtre (F2) dans son entier de manière à rendre accessible au pointeur l'ensemble de ladite seconde fenêtre.

Des aspects préférés, mais non limitatifs, du procédé selon l'invention sont les suivants :

- ladite autre condition est déterminée par le fait que ledit pointeur atteint un bord de la première fenêtre alors que ledit pointeur se situait dans la zone de la seconde fenêtre non recouverte par la première fenêtre.

- l'étape d'affichage de la seconde fenêtre dans son entier comprend une sous-étape de déplacement de l'affichage de la première fenêtre pour qu'elle ne recouvre plus ladite seconde fenêtre.

- le déplacement de l'affichage de la première fenêtre est corrélé aux déplacements du pointeur après qu'il a atteint le bord de la première fenêtre.

- le procédé comprend en outre une étape consistant à effectuer une signalisation auprès de l'utilisateur lors de l'atteinte par le pointeur du bord de la première fenêtre.

- ladite signalisation est choisie dans le groupe comprenant une signalisation visuelle, une signalisation sonore et une signalisation tactile telle qu'un retour de force au sein d'un périphérique d'entrée manuel qui commande ledit pointeur.

- ladite autre condition est déterminée par le fait que ledit pointeur a séjourné dans la zone de la seconde fenêtre non recouverte par la première fenêtre pendant une durée supérieure à un seuil prédéterminé.

- ladite autre condition est déterminée par le fait qu'une touche d'un clavier du poste informatique est actionnée.

- ladite autre condition est déterminée par la réception dans le poste informatique d'une commande vocale.

- le procédé comprend en outre l'étape additionnelle :

(c) lorsque ledit pointeur a séjourné dans la zone de la seconde fenêtre qui était précédemment recouverte par la première fenêtre et qu'au moins une seconde autre condition est remplie, ré-affichage de la première fenêtre (F1) de manière qu'elle recouvre partiellement la seconde fenêtre comme précédemment.

- la première autre condition est le franchissement par le curseur du bord de la première fenêtre tandis que la seconde autre condition est un déplacement inverse du pointeur.

Selon un autre aspect de l'invention, dans le cas où le procédé est mis en œuvre sur un poste informatique client, les contenus des fenêtres étant aptes à être chargés à partir d'un poste informatique serveur, le contenu de la première fenêtre est à chargement rapide et le contenu de la seconde fenêtre est à chargement plus lent:

- le contenu de la première fenêtre est affiché dès après son chargement,
- l'étape de détection est démarrée seulement à la fin du chargement du contenu de la seconde fenêtre.

D'autres aspects, buts et avantages de la présente invention apparaîtront mieux à la lecture de la description détaillée suivante d'une forme de réalisation préférée de celle-ci, donnée à titre d'exemple non limitatif et faite en référence aux dessins annexés, sur lesquels

les figures 1 à 13 illustrent schématiquement différents états et comportements d'un procédé de gestion d'affichage selon l'invention,

les figures 14 à 17 sont des vues schématiques en perspective illustrant le comportement de deux éléments d'affichage superposés et des zones associées selon une généralisation de l'invention,

la figure 18 est une vue en plan des différentes zones exploitées,

la figure 19 représente deux positions possibles d'un élément d'affichage donné,

la figure 20 indique une symbolique de représentation des états du système de gestion d'affichage,

la figure 21 est un diagramme d'états-transitions du système,

la figure 22 illustre par une modélisation en section, transversalement au plan de l'affichage, les comportements illustrés sur la figure 21,

les figures 23 à 25 sont des vues schématiques en perspective illustrant le comportement du système de l'invention avec trois éléments d'affichage superposés,

la figure 26 illustre par une modélisation en section, transversalement au plan de l'affichage, le comportement d'un système selon l'invention appliqué à une pluralité d'éléments d'affichage superposés.

les figures 27 à 31 illustrent les différentes étapes du comportement d'un procédé de gestion d'affichage selon l'invention, appliqué à des fenêtres d'un système d'exploitation,

la figure 32 illustre une structure de liens entre pages ajoutés par un utilisateur selon la présente invention,

la figure 33 illustre la combinaison de la Toile et d'une toile personnelle constituée par de tels liens ajoutés,

la figure 34 illustre schématiquement la façon de créer l'association entre une page et ses liens ajoutés,

la figure 35 illustre une façon d'afficher une page et les liens ajoutés qui y ont été associés,

les figures 36a à 36c illustrent un exemple de combinaison de liens ajoutés,

la figures 37 illustre différentes opérations permettant de créer des liens ajoutés entre pages,

les figures 38 et 39 illustrent l'application d'un mécanisme de gestion d'affichage selon l'invention à une superposition de pages pourvues de leurs liens ajoutés,

la figure 40 illustre un agencement d'affichage différent des liens ajoutés associés à des pages,

la figure 41 illustre la correspondance entre les opérations de création de liens ajoutés et le stockage de tels liens ajoutés,

la figure 42 illustre de façon analogue à la figure 41 une correspondance dans le cas où les liens ajoutés incluent également des liens ajoutés inverses,

la figure 43 illustre schématiquement un mode de stockage des liens ajoutés

la figure 44 illustre schématiquement un premier mode d'obtention via réseau d'une page et des liens ajoutés qui y ont été associés,

la figure 45 illustre schématiquement un deuxième mode d'obtention via réseau d'une page et des liens ajoutés qui y ont été associés,

les figure 46 à 51 illustrent six exemples de manipulations de pages et de liens ajoutés dans le cadre de pages contenues dans des répertoires d'utilisateur,

les figures 52 et 53 illustrent deux modes de présentation de liens mémorisés ou liens ajoutés de deuxième rang en relation avec une page courante

la figure 54 illustre un exemple d'interface d'accès à l'utilisateur dans le cadre d'une page à laquelle sont associés des liens ajoutés,

la figure 55a illustre un exemple d'interface utilisateur dans laquelle à une page sont associés d'une part des liens ajoutés, et d'autre part une sélection de répertoires proches,

la figure 55b illustre un détail d'affichage d'une page et de liens ajoutés associés permettant de changer un attribut d'un lien ajouté,

la figure 56 illustre dans une représentation standard UML un diagramme de classe pour les liens ajoutés et leur attribut précité,

la figure 57 présente de manière schématique les principes d'accès direct aux liens ajoutés et aux répertoires,

les figures 58 et 59 illustrent schématiquement les fondements du mode de calcul primaire de proximité entre répertoires pour une page courante donnée,

la figure 60 illustre schématiquement les fondements du mode de calcul de proximité transitive entre répertoires pour une page courante donnée

les figures 61 à 63 illustrent schématiquement les fondements de trois modes évolués de calcul de proximité entre répertoires,

la figure 64 illustre deux modes de représentation graphique d'un lien ajouté associé à une page,

la figure 65 illustre schématiquement l'accès direct aux répertoires ,

la figure 66 illustre un détail d'affichage d'un répertoire et des liens qu'il contient permettant de changer un attribut d'un lien,

la figure 67 illustre dans une représentation standard UML un diagramme de classe pour les liens et leur attribut précité,

la figure 68 illustre schématiquement le placement de contenants dans un document,

la figure 69 illustre schématiquement l'importation d'un contenu d'un contenant dans un autre,

la figure 70 illustre schématiquement la dérivation d'un contenant,

la figure 71 illustre schématiquement l'importation d'un contenu d'un document vers un contenant d'un autre document,

la figure 72 illustre schématiquement le principe de composition des contenus,

la figure 73 illustre schématiquement le principe de structuration de références entre éléments,

la figure 74 illustre schématiquement l'accès indirect à une page sur la Toile par l'intermédiaire du système,

la figure 75 illustre schématiquement le procédé mis en œuvre par le système,

la figure 76 illustre schématiquement deux contenants,

la figure 77 illustre schématiquement le procédé d'importation,

la figure 78 illustre schématiquement le procédé d'importation d'un contenu dans un contenant de catégorie différente,

la figure 79 illustre schématiquement des containers imbriqués,

les figures 80a et 80b illustrent schématiquement le procédé de dérivation,

la figure 81 illustre schématiquement une structure infinie de dérivation,

la figure 82 illustre schématiquement un exemple simple de structure de document,

la figure 83 illustre schématiquement le principe de stockage des attributs des contenus,

la figure 84a illustre un exemple d'interface utilisateur dans laquelle à une page sont notamment associés des liens ajoutés et des liens sur des contenus dans un contenant,

la figure 84b illustre un exemple d'interface utilisateur dans laquelle figurent des cases à cocher « Demandeur » et « Offreur »,

la figure 85 est un diagramme de classe en représentation UML d'une structure hiérarchique de Contenants d'informations utilisée selon un troisième aspect de la présente invention,

la figure 86 est un diagramme de classe incluant la notion de Profil d'Utilisateur,

la figure 87 est une autre illustration de l'organisation de Profils, Contenants et Contenus,

la figure 88 est un diagramme de classe dans lequel sont introduites les notions de Référence à Contenu (lien vers contenu) et de popularité de Contenu,

la figure 89 est un diagramme de classe dans lequel sont introduites les notions de Contenu Direct et de Contenu Indirect,

la figure 90 illustre un avantage essentiel d'une forme de réalisation de base de l'invention,

la figure 91 est un diagramme de classe illustrant la notion de Référence en cascade,

la figure 92 illustre un principe de propagation de profil,

les figures 93 et 94 illustrent la création de liens correspondant à la propagation de Profils de la figure 92,

la figure 95 illustre un autre principe de propagation de profil,

les figures 96 et 97 illustrent la création de liens correspondant à la propagation de Profils de la figure 95,

les figures 98 et 99 illustrent des créations de liens occasionnées par un processus de Référencement,

la figure 100 est un diagramme de classe intégrant les notions de propagation de Profils et de Référencement de Profils,

les figures 101 à 104 illustrent quatre configurations possibles de transition de liens,

les figures 105 à 107 illustrent des créations de liens concernant l'archivage d'éléments et leur restauration,

la figure 108 illustre un principe de collecte de Profils sous forme d'un ensemble de fichiers communs,

la figure 109 est un diagramme de classe mettant en œuvre le principe de la figure 108,

la figure 110 illustre une construction d'arborescence dans le cadre du principe de la figure 108,

la figure 111 est un diagramme de classe reprenant l'ensemble des principes des figures 85 à 110,

la figure 112 illustre une structure arborescente de classeurs selon la présente invention,

la figure 113 est un diagramme de classe intégrant cette notion de classeurs,

la figure 114 illustre la structure d'une base de données SQL associée à l'organisation en classeurs,

la figure 115 illustre une table de classeurs,

la figure 116 illustre une table d'éléments propres à l'utilisateur dans un classeur,

la figure 117 illustre une table de liens,

la figure 118 illustre la création d'un élément « Dérivé »,

la figure 119 illustre la suppression d'un sous-élément propre non Dérivé,

les figures 120a à 120c illustrent la suppression d'un sous-élément propre Dérivé,

la figure 121 illustre une table de liens lors de la restauration d'un sous-élément,

la figure 122 illustre le comportement de sous-éléments « implicites »,

la figure 123 illustre une table de liens correspondante,

la figure 124 illustre l'acceptation de sous-éléments implicites,

la figure 125 illustre une table de liens correspondante,

la figure 126 illustre le refus d'un sous-classeur implicite,

la figure 127 illustre la table de liens correspondante,

la figure 127a illustre un exemple possible d'interface utilisateur,

la figure 128 illustre une mesure de proximité entre deux Profils,

la figure 129 illustre une autre mesure de proximité possible,

la figure 130 illustre les contenus respectifs de deux pages de classeurs d'utilisateurs, organisées selon deux critères communs,

la figure 131 illustre des suggestions par Recommandation Collaborative,

la figure 132 illustre les deux pages de classeurs d'utilisateurs après insertion d'un élément accepté,

la figure 133 illustre ces deux mêmes pages de classeurs, avec des insertions d'éléments acceptés en des positions répondant à d'autres critères,

la figure 134 illustre une architecture de serveurs permettant, avec le présente invention, de mettre en œuvre des fonctions d'anonymat et de commission sur opérations de commerce électronique.

la figure 135 schématise le comportement séquentiel d'une pluralité d'objets, selon un quatrième aspect de la présente invention,

la figure 136 illustre la notation normalisée « UML »,

la figure 137 illustre un exemple simplifié de diagramme de classe,

la figure 138 illustre le diagramme d'état-transition pour l'exemple précité,

la figure 139 représente un objet simplifié comportant une méthode,

la figure 140 illustre par un exemple simplifié une équivalence entre attribut d'une classe et lien entre deux classes,

la figure 141 illustre l'équivalence entre une action sur transition et un état intermédiaire à transition sortante automatique,

la figure 142 illustre l'ajout d'un pseudo-état à un diagramme d'état-transition standard,

la figure 143 illustre un exemple simplifié d'un objet possédant deux types d'actions,

la figure 144 illustre un exemple enrichi de l'exemple simplifié de la figure 137,

la figure 145 illustre le comportement d'un objet de la figure 144,

la figure 146 illustre le principe de décomposition des états d'un objet,

la figure 147 donne un exemple d'un objet auquel est appliqué le principe illustré sur la figure 146,

la figure 148 illustre l'objet après transformation selon ce principe,

la figure 149 illustre la structure d'un message émis par l'objet,

la figure 150 illustre le comportement de base d'un séquenceur en réponse à un tel message,

la figure 151 illustre la notion de cycle utilisée selon l'invention,

la figure 152 illustre le comportement d'objets sous contrôle du séquenceur pour certains types de cycles,

la figure 153 illustre par un diagramme d'état-transition le comportement du séquenceur avec le contrôle illustré sur la figure 152,

la figure 154 reprend la figure 148 avec un enrichissement,

la figure 155 illustre le même enrichissement dans le cas où un objet possède un certain type d'actions,

la figure 156 illustre une succession de messages mémorisés par le séquenceur,

la figure 157 illustre le diagramme d'état-transition du séquenceur avec une fonction de gestion d'historique,

la figure 158 illustre une exploitation additionnelle par le séquenceur des messages de la figure 156,

la figure 159 illustre par un diagramme d'état-transition le comportement du séquenceur modifié en conséquence,

la figure 160 illustre le diagramme d'état-transition complet du séquenceur,

la figure 161 illustre les diagrammes d'état-transition initiaux de six objets dans un exemple de mise en œuvre de l'invention,

la figure 162 indique les nouveaux diagrammes d'état-transition des objets selon l'aspect de l'invention illustré jusqu'ici,

la figure 163 illustre le diagramme de séquence de certains événements et actions dans une certaine phase pour ce même exemple,

la figure 164 illustre un extrait de la liste de messages correspondante,

la figure 165 illustre le diagramme de séquence dans une autre phase,

la figure 166 illustre un extrait correspondant de la liste de messages,

la figure 167 illustre le diagramme de séquence dans une troisième phase,

la figure 168 illustre un extrait correspondant de la liste de messages,

la figure 169 illustre le diagramme de séquence d'un autre type d'action géré par le séquenceur,

la figure 170 illustre un exemple d'objets et de leurs liens,

la figure 171 illustre une structure arborescente d'objets et de liens,

la figure 172 illustre de façon plus détaillée une telle structure arborescente,

la figure 173 illustre un chemin entre objets utilisé selon un cinqième aspect de l'invention,

la figure 174 illustre une composition particulière de liens dans un chemin,

la figure 175 illustre le principe de regroupement de certains objets d'une structure arborescente,

la figure 176 illustre l'application d'un tel regroupement à des zones de visualisation associées à des objets,

la figure 177 illustre des liens en relation avec la proximité entre zones de visualisation,

la figure 178 illustre des regroupements d'objets selon le même principe, avec une granularité croissante,

la figure 179 illustre deux possibilités de transformations de liens,

la figure 180 illustre par un diagramme UML simplifié deux types d'objets, à savoir objet proprement-dit ou élément de collection,

les figures 181 et 182 illustrent une première phase d'un procédé de connexion entre objets,

la figure 183 illustre sous une autre forme un regroupement d'objets,

la figure 184 illustre un certain nombre d'actions dans une structure arborescente d'objets en liaison avec des objets d'affichage regroupés sur un objet constituant une zone de visualisation,

la figure 185 illustre le diagramme de séquence correspondant,

les figures 186 à 189 illustrent une application du cinquième aspect de l'invention à des liens de composition changeants,

la figure 190 illustre une liste de messages de séquenceur conforme au cinquième aspect de l'invention dans le contexte des figures 186 à 189,

la figure 191 illustre une transition d'objet dans le cadre d'un perfectionnement du cinquième aspect de l'invention,

la figure 192 illustre le principe d'une connexion anticipative entre objets,

la figure 193 illustre une telle connexion anticipative sous la forme d'un diagramme UML,

la figure 194 illustre le diagramme de séquence correspondant,

les figures 195 et 196 illustrent deux modes de connexion anticipative d'objets, en fonction du type d'objet,

la figure 197 illustre l'architecture de base d'un serveur de données partagées et de postes clients,

la figure 198 illustre la manipulation d'objets partagés par deux utilisateurs de postes clients,

la figure 199 illustre un principe de réplication mixte selon un sixième aspect de l'invention,

la figure 200 illustre le même principe, appliqué au cas où les objets partagés sont gérés par deux serveurs,

la figure 201 illustre les diagrammes d'état-transition pour un exemple de quatre objets, dont trois sont de types différents,

la figure 202 illustre le comportement de ces quatre objets lors d'une transition causée par l'utilisateur sur l'un des objets,

la figure 203 illustre le comportement associé au niveau de deux listes de messages, respectivement du côté d'un poste client et du côté d'un serveur,

la figure 204 illustre une codification d'objets utilisée sur la figure 205,

la figure 205 illustre la propagation d'une transition à partir d'un poste client vers un serveur et vers d'autres postes clients,

la figure 206 illustre le même principe de propagation dans le cas où il existe deux serveurs,

la figure 207 est un schéma de l'architecture des moyens de communication entre les postes clients et le serveur,

la figure 208 est un diagramme de séquence d'une procédure de connexion entre poste client et serveur,

la figure 209 est un schéma analogue à celui de la figure 207, illustrant les flux d'informations lors d'un appel de méthode,

les figures 210 et 211 sont des diagrammes de séquence correspondants,

la figure 212 est un diagramme de séquence illustrant la fin d'une connexion,

les figures 213 et 214 sont des diagrammes de séquence illustrant l'attribution de verrous à des objets,

les figures 215 et 216 sont des diagrammes d'état-transition illustrant les comportements de verrous sur un objet,

les figures 217 et 218 sont des diagrammes de séquence illustrant une demande de verrou de lecture et d'écriture d'un poste client à un serveur,

la figure 219 est une autre forme de diagramme de séquence illustrant un temps de latence dans le processus d'attribution du verrou,

la figure 220 est un diagramme de séquence illustrant une demande de verrou de lecture d'un poste client à un serveur,

la figure 221 est une autre forme de diagramme de séquence illustrant un temps de latence dans le processus d'attribution du verrou,

les figures 222 à 224 illustrent une situation de blocage mutuel lors de l'attente de verrous entre objets,

la figure 225 illustre un exemple de demandes et attributions de verrous sur les axes des temps logiques de deux postes clients en association avec un tel blocage mutuel,

la figure 226 illustre deux situations correspondant respectivement à un blocage mutuel et à l'absence d'un tel blocage,

la figure 227 illustre le recours à un compteur pour délier des situations de blocage mutuel,

la figure 228 illustre des propagations de transitions entre cinq objets, afin d'illustrer une fonction de retour en arrière,

la figure 229 illustre l'état de listes de messages correspondants côté serveur et côté client,

la figure 230 illustre d'autres propagations de transitions dans le même exemple,

la figure 231 illustre l'état correspondant de la liste de messages côté serveur,

la figure 232 illustre d'autres propagations de transitions encore dans le même exemple,

la figure 233 illustre l'état correspondant de la liste de messages côté client,

la figure 234 est un diagramme d'états et de messages illustrant une situation de conflit entre serveur et client,

les figures 235 à 238 illustrent au niveau de la liste des messages côté serveur les étapes de résolution du conflit.

Introduction de la description

On va présenter dans la suite dans six chapitres, différents systèmes et procédés - liés entre eux - visant d'une façon générale des perfectionnements à des outils informatiques liés à l'Internet et plus généralement à l'informatique partagée via réseau.

Le chapitre I vise en particulier des perfectionnements à la gestion de l'affichage sur écran d'éléments graphiques tels que des pages de la Toile, des parties de pages ou encore des fenêtres d'un système d'exploitation, dans différentes circonstances telles que le chargement de pages ou des opérations de glisser-déposer.

Le chapitre II vise, en utilisant notamment la gestion d'affichage du chapitre I, à permettre à un utilisateur de créer des liens personnels entre des pages de la Toile, et vise également différentes techniques pour mettre à profit ces liens créés.

Le chapitre III vise, en utilisant notamment les techniques de liens créés du chapitre II, différentes techniques d'enrichissement mutuel des connaissances notamment via l'Internet. Il reprend à cet égard des notions exposées dans le chapitre II.

Le chapitre IV concerne quant à lui les applications orientées objet et des moyens destinés à rendre déterministe le comportement de telles applications lorsqu'elles sont partagées notamment via l'Internet.

Le chapitre V vise, dans la lignée des applications partagées du chapitre IV, à proposer des techniques d'exécution telles applications qui s'affranchissent au moins en partie des problèmes de bande passante faible ou limitée.

Le chapitre VI concerne enfin, toujours dans le cadre d'applications orientées objet partagées et en liaison particulière avec les chapitres IV et V, des techniques permettant de faciliter la bonne exécution partagée d'applications possédant différents types d'objets.

Chapitre I - Gestion de zones d'affichage superposées (Figures 1-27)

En référence tout d'abord à la figure 1, on a représenté schématiquement une page interactive P, telle qu'une page en langage HTML, qui doit être chargée sur un réseau lent et/ou irrégulier tel que l'Internet, et qui est volumineuse (quelques dizaines de Kilo-octets, voire plus de cent Kilo-octet, ou même plus).

Cette page est décomposée en deux zones d'affichage Z1 et Z2 couvrant ensemble, typiquement, une fenêtre d'un logiciel de navigation sur Internet. Dans l'exemple de la figure 1, un élément de page E2 est chargé dans la zone Z1, tandis qu'un élément de page E5 est chargé dans la zone Z2.

L'une de ces zones (la zone Z1 en l'occurrence) est définie comme pouvant « tolérer » le recouvrement de l'affichage de la page P par l'affichage d'éléments externes.

Autrement dit, la décomposition en zones permet de distinguer une zone (Z1) dans laquelle l'affichage de l'élément courant (l'élément E2) peut être recouvert (entièrement ou partiellement) par l'affichage d'un (ou plusieurs) éléments qui n'appartiennent pas à la page interactive P.

Bien entendu, les zones en question pourraient être de n'importe quelle forme, et constituées de parties (sous-zones) contiguës ou non.

Selon la présente invention, on vise à faire patienter l'utilisateur pendant le chargement de la page interactive, l'idée de base consistant à lui présenter au préalable un élément supplémentaire (élément E1 dans les schémas qui suivront) dont la caractéristique essentielle est d'être légère en termes de volume de données à transmettre, en étant par exemple constituée d'un simple texte introductif, c'est-à-dire très rapidement chargeable.

La figure 2 illustre l'élément E1 affiché dans la zone Z1 après avoir été chargé.

L'affichage de cet élément E1 est effectuée dans la zone Z1 afin de pouvoir recouvrir par la suite l'affichage de certains éléments de la page interactive (et plus précisément l'élément E2) quand ceux-ci seront chargés.

On va maintenant décrire un certain nombre de détails de mise en œuvre concrète de l'invention.

Section 1- Stabilité de l'affichage de l'élément E1

La durée de chargement cumulée des éléments E2 et E5 destinés à constituer la page P n'étant pas exactement prévisible, et par ailleurs la durée nécessaire à l'examen complet par l'utilisateur du contenu de l'élément E1 - c'est-à-dire la durée de *lecture* du texte qu'il contient - étant dépendante des capacités et de l'attention de l'utilisateur, il est avantageux que l'élément E1 reste affiché jusqu'à que l'utilisateur l'ôte de sa propre initiative. Il est en effet désagréable pour l'utilisateur de se voir tout à coup privé d'un texte il est en train de lire.

L'approche adoptée, qui va maintenant être décrite plus en détail en référence à la figure 3, est la suivante : dès la fin du chargement des éléments E2 et E5 initialement nécessaires à l'exécution de la page interactive P, tout ou partie de la zone Z1 dans laquelle doit être affiché l'élément E1 sert de « zone sensible », de manière à ce que, quand le pointeur d'une souris (ou tout autre périphérique d'entrée de la machine constituant un dispositif de pointage) entre dans cette zone Z1 (et sans qu'un clic de souris ne soit nécessaire), l'affichage de l'élément E1 est automatiquement remplacé par celui de l'élément E2.

On observera ici que l'on peut équiper la machine de moyens (bouton, fenêtre de dialogue ou autre) permettant alors à l'utilisateur de déclarer le cas échéant que par la

suite, il ne souhaite plus voir l'élément E1 affiché. Ce moyen peut par exemple être un sous-élément d'interactivité dans E2 ou E1.

Section 2- Ré-affichage de l'élément E1

Sauf indication contraire explicitement donnée par l'utilisateur comme indiqué ci-dessus, on prévoit alors que l'utilisateur puisse retrouver ultérieurement l'élément E1 de sa propre initiative et le plus facilement possible, par exemple dans le cas où il l'aurait ôté par erreur.

Or, après le remplacement de l'élément E1 par l'élément E2, puisque l'élément E2 doit offrir à l'utilisateur toutes ses propres possibilités d'interactivité (liens hypertexte, etc.) dans la zone d'affichage Z1, il est avantageux que cette zone Z1 ne serve pas à déclencher l'opération inverse, c'est-à-dire le remplacement de l'élément E2 par l'élément E1.

Pour satisfaire à cette contrainte, et comme illustré sur la figure 4, au moment où l'élément E2 est affiché, on prévoit que tout ou partie de la zone d'affichage Z2 dans laquelle est affiché l'élément E5 devienne à son tour « zone sensible » ; l'élément E1 pourra ainsi être ré-affiché dès la détection de l'entrée du pointeur dans cette zone.

Toutefois, dès que l'élément E1 a été de nouveau affiché, il faut que l'élément E5 offre toutes ses propres possibilités d'interactivité. C'est alors, comme précédemment, tout ou partie de la zone Z1 (contenant alors l'élément E1) qui redevient zone sensible. Ceci est illustré sur la figure 5.

Les comportements des figures 4 et 5 peuvent, dans la présente forme de réalisation, être enchaînés à l'infini, aussi longtemps que l'utilisateur n'a pas appelé d'autres pages.

Bien évidemment, tant E2 que E1 peuvent aussi posséder des sous-éléments d'interaction, constituant d'autres zones sensibles, pour respectivement amener et enlever l'élément E1.

On va maintenant décrire en détail différents déroulements du procédé selon l'invention.

Cas 1 : Le pointeur de la souris se trouve initialement dans la zone Z2

Il s'agit du cas où, au moment où les éléments nécessaires à l'exécution initiale de la page interactive P sont finis d'être chargés, le pointeur de la souris se trouve dans la zone d'affichage qui ne contient pas l'élément E1.

Le déroulement du procédé est alors le suivant :

- l'élément E1 est chargé et affiché en premier, et reste affiché tant que les éléments nécessaires à l'exécution initiale de la page interactive P (à savoir les éléments E2 et E5) ne sont pas encore chargés ;

- l'élément E1 reste affiché, même après chargement des éléments E2 et E5, tant que le pointeur actionné par la souris n'effectue pas une entrée dans la zone d'affichage Z1 contenant l'élément E1 ;

- c'est seulement lors d'une telle entrée qu'il est remplacé par les éléments E2 et E5 ;

- au cas où le pointeur est ensuite amené dans la zone d'affichage Z2 qui ne contient pas l'élément E2, l'élément E1 est affiché à nouveau (E2 peut rester affiché ou pas).

Cette séquence est présentée dans le diagramme de la figure 6.

Cas 2 : Le pointeur de la souris se trouve initialement dans la zone Z1

Il s'agit du cas où, au moment où les éléments E2 et E5 nécessaires à l'exécution initiale de la page interactive P ont été chargés, le pointeur de la souris se trouve dans la zone d'affichage Z1 qui contient l'élément E1.

Dans ce cas l'élément E1 reste affiché aussi longtemps que le pointeur n'est pas sorti de la zone Z1, puis à nouveau entré dans celle-ci ; c'est seulement dans ce dernier cas que l'élément E1 est remplacé par les éléments E2 et E5 sur l'écran d'affichage.

Cette séquence est présentée dans le diagramme de la figure 7.

Approche dite de « streaming »

Bien entendu, dans tous les cas, les éléments E2 et E5 de la page interactive P pourront être transmis dans une approche dite de « streaming », approche classique dans laquelle des éléments d'une page sont affichés et exécutés avant que la page soit entièrement chargée. Il n'est en effet pas nécessaire d'attendre que la totalité de ces éléments de la page soient chargés pour activer la ou les « zones sensibles ».

On va présenter ci-après une approche de « streaming » pour l'élément E1, qui dans le présent exemple est constitué par deux sous-éléments E1.1 et E1.2.

Plutôt que d'attendre la fin du chargement complet de l'élément E1, les parties distinctes E1.1 et E1.2 de l'élément E1 (qui peuvent être par exemple principalement du texte) peuvent être chargées et affichées successivement, dans l'ordre de lecture.

Le principe d'affichage des éléments en fonction des déplacements du pointeur de la souris peut rester identique, comme le présente le schéma de la figure 8.

En variante, le comportement des différentes parties de l'élément E1 peuvent être dissociées, selon les règles énoncées ci-dessous et décrites en référence aux figures 9 à 12, sur lesquelles chaque « fenêtre » a pour but de montrer d'une part les nouveaux éléments affichés en réaction à l'action de déplacement du pointeur de la souris présentée dans la fenêtre immédiatement précédente, le cas échéant, d'autre part une

nouvelle action de déplacement de la souris dont l'effet est montré dans la fenêtre suivante, le cas échéant.

Règle 1

Il s'agit ici d'un simple rappel du principe déjà énoncé dans le paragraphe « *Cas 2 : Le pointeur de la souris se trouve initialement dans Z1* » :

Dans le cas où, au moment de la fin du chargement des éléments E2 et E5 de la page interactive P, le pointeur de la souris se trouve dans la zone Z1, les déplacements du pointeur n'ont pas d'effet tant que le pointeur n'est pas amené en dehors de la zone Z1 (c'est-à-dire introduit dans la zone Z2) puis retourné ensuite dans la zone Z1. Ceci est illustré sur la figure 9.

Règle 2

Le déplacement du pointeur de la souris, de la zone Z2 à la zone Z1, déclenche le remplacement de l'affichage du seul sous-élément touché par le sous-élément correspondant de la page interactive. Dans l'exemple illustré sur la figure 10, l'entrée du pointeur dans la région affectée au sous-élément E1.1 de l'élément E1 provoquera son remplacement par un sous-élément E2.1 de l'élément E2, tandis que l'autre sous-élément E1.2 de l'élément E1 restera intact.

Règle 3

Cette règle aborde la réversibilité du comportement défini par la règle 2 : si le pointeur revient ensuite dans la zone Z2, même en étant passé par d'autres régions entre-temps, mais à condition d'avoir automatiquement défait les remplacements causés entre-temps, l'affichage de l'élément E2.1 est remplacé par celui de l'élément E1.1. Ceci est illustré sur la figure 10.

Les schémas des figures 11 et 12 présentent d'autres exemples de mise en œuvre de la réversibilité, tels que décrits ci-dessous :

Règle 4

Le déplacement du pointeur de la souris de la région occupée par le sous-élément E2.1 à la région occupée par le sous-élément E1.2 déclenche le remplacement de l'affichage du sous-élément E1.2 par celui du sous-élément correspondant E2.2 de la page interactive.

Règle 5

Cette règle décrit une forme de réversibilité de la règle 4 : si le pointeur retourne de la région du sous-élément E2.2 à à la région du sous-élément E2.1, même en étant passé par d'autres régions entre-temps mais à condition d'avoir automatiquement défait les remplacements causés entre-temps, l'affichage du sous-élément E2.2 est remplacé par celui du sous-élément E1.2. Ceci est illustré sur la figure 11.

Le schéma de la figure 12 présente une variante de mise en œuvre de la règle 5.

Supposons qu'après le parcours suivant du pointeur de la souris :

Z2 → E1.1 (qui devient E2.1) →E1.2 (qui devient E2.2)

le pointeur de la souris soit amené de la région occupée par le sous-élément E2.2 jusque dans la zone Z2. Ceci, du fait qu'il ne s'agit pas un retour au sens de la Règle 3, n'engendre pas le remplacement du sous-élément E2.2 par le sous-élément E1.2.

Règle 6

Cette règle consiste à rendre l'opération ci-dessus réversible : si le pointeur retourne de la zone Z2 à la région occupée par le sous-élément E2.2, même en étant passé par d'autres zones entre-temps mais à condition d'avoir automatiquement défait les remplacements causés entre-temps et si le retour s'effectue avant un « seuil temporel » donné, l'opération est considérée comme étant défaite. Ceci est illustré sur la figure 12. Cette forme de réalisation permet avantageusement de ne plus appeler les sous-éléments constituant l'élément initial E1 lorsqu'une période de temps prédéterminée s'est écoulée pendant laquelle l'utilisateur est resté sur les sous-éléments de l'élément E2, ce qui est révélateur par exemple d'un début de travail de l'utilisateur sur l'élément E2.

Section 3 - Exemples d'applications du présent chapitre

Dans le cadre d'applications Internet ou autres, le procédé selon l'invention peut être complété par exemple par des éléments présentés dans le diagramme de la figure 13. Les informations portés par les éléments qui y sont représentés sont par exemple les suivantes :

Elément E1 : texte (et le cas échéant image et/ou son, etc) introductif ou d'accueil, l'essentiel étant que sont chargement soit relativement rapide comparativement aux éléments E2 et E5 (constituant par exemple la page principale du site) ;

Elément E3, occupant initialement la zone Z2 : par exemple un message du type « Prière de patienter ; la page interactive est en train d'être chargée » ;

Elément E4, remplaçant l'élément E3 dans la zone Z2 : par exemple un message du type « Le chargement minimal de la page interactive est terminé » ;

Elément E2 : la partie de la page principale occupant la zone Z1 ;

Elément E5 : la partie de la page principale occupant la zone Z2.

L'homme du métier saura aisément mettre en œuvre l'invention dans une machine par exemple du type ordinateur personnel (au standard « PC », « MacOS » (marque déposée), etc.), comportant un processeur, des mémoires, et entrées/sorties, un écran d'affichage et une liaison avec un serveur délocalisé, ceci par exemple par programmation de routines appropriées en liaison avec le protocole de chargement des éléments depuis le serveur, sachant notamment que

- la définition de « zones sensibles » pour un pointeur de souris ou analogue
- la détection de la présence du pointeur dans ces zones,
- la détection de fin de chargement de différentes pages HTML ou de leurs éléments individuels (textes, images, sons, etc.)

constituent des outils classiques à disposition des développeurs.

On peut par ailleurs apporter de nombreuses variantes à la présente invention.

En premier lieu, on peut prévoir que la région dans laquelle va s'afficher l'élément initial E1 ne soit pas superposée exactement à la région dans laquelle va s'afficher l'élément E2 de la page principale. En particulier, la région occupée par l'élément E1

peut être plus petite que la région (zone Z1) dans laquelle s'affichera, dans les conditions définies plus haut, l'élément E2. Dans ce cas, la partie de la zone Z1 non recouverte par la région occupée par l'élément E1 peut par exemple rester vierge aussi longtemps que l'élément E2 n'est pas affiché. Dans cette hypothèse, c'est avantageusement l'entrée du pointeur de la souris dans la seule région occupée par l'élément E1 qui provoquera le remplacement de l'élément E1 par l'élément E2. La région occupée par l'élément E1 peut aussi être plus grande que la région (zone Z1) dans laquelle s'affichera l'élément E2.

Selon une autre variante, on peut prévoir que le rappel de l'affichage de l'élément E1 en lieu et place de l'élément E2 s'effectue non pas en amenant le pointeur de la souris dans la zone Z2, mais par toute autre action et notamment :

- en amenant le pointeur de la souris (avec le cas échéant la nécessité d'un clic) dans un bouton, onglet, etc. affiché au voisinage de l'élément E2 lorsque ce dernier est affiché (ou encore appartenant à l'élément E2) ;
- par appui sur une touche spécifique du clavier de la machine,
- etc.

On peut par ailleurs prévoir que les différents sous-éléments de l'élément initial E1 soient contigus ou non contigus.

Au surplus, on peut recourir à différentes techniques de déplacement d'éléments pour retirer E1 de la zone d'affichage Z1 (ou de la ou des régions qu'il occupe dans cette zone, le cas échéant) avec un effet visuel donné pour l'utilisateur. En particulier, on peut prévoir une translation horizontale ou verticale progressive de la représentation de l'élément E1 vers l'extérieur de l'écran.

Egalement, on peut prévoir que l'élément E1, après avoir été initialement affiché, soit recouvert par certains sous-éléments (boutons, etc.) appartenant à l'élément E2 en cours de chargement (cas par exemple d'un chargement de E2 en « streaming »). l'élément E1 est donc en quelque sorte en « sandwich » entre certains sous-éléments

de l'élément E2 (ceux qui resteront invisibles jusqu'à ce que l'élément E1 soit ôté) et d'autres sous-éléments de l'élément E2 qui vont se comporter comme indiqué ci-dessus.

Dans ce cas, la disparition de l'élément E1 lors de l'entrée du pointeur dans la zone associée, lorsqu'elle se fait par translation vers l'extérieur de l'écran de la représentation à l'écran dudit élément, est avantageusement effectuée en laissant immobiles le ou les sous-éléments de l'élément E2 alors visibles à l'écran.

On peut enfin prévoir que l'élément E1 affiché initialement contienne des sous-éléments d'interactivité (liens vers l'autres pages, boutons déclenchant des actions, etc.). Dans ce cas, on prévoit avantageusement que ces sous-éléments soient actifs (c'est-à-dire actionnables par pointeur de souris notamment) au moins pendant la période au cours de laquelle l'élément E2 est en cours de chargement, et que ces sous-éléments deviennent inactifs dès le moment où l'entrée du pointeur dans la zone de l'élément E1 doit provoquer la disparition de cet élément. En effet, les sous-éléments d'interactivité en question ne sont alors plus atteignables.

On va maintenant décrire en référence aux figures 14 à 24 une généralisation de la présente invention telle que décrite ci-dessus.

On se place tout d'abord dans le situation où l'on souhaite visualiser deux pages (par exemple des pages HTML) dans une seule fenêtre d'affichage.

Pour bien faire comprendre cet aspect de l'invention, on va considérer que ces deux pages sont dans deux plans différents, la page P1 du plan supérieur recouvrant partiellement la page P2 du plan inférieur, en étant entièrement visible dans la fenêtre de visualisation (dont la taille est ici légèrement supérieure à celle de la page P2), comme illustré sur la figure 14.

La page P1 du plan supérieur peut également occuper une position décalée; elle recouvre encore partiellement la page P2, mais n'est visible que partiellement, marginalement, comme l'illustre la figure 15.

Dans cette forme de réalisation, on prévoit en outre une « pince » virtuelle qui peut fixer ensemble les deux pages, comme on le détaillera plus loin, et ce dans une ou l'autre de leurs deux positions mutuelles. Cette pince est schématisée par PC sur les figures 16 et 17.

En vue en plan, l'utilisateur peut ainsi distinguer quatre zones Z11, Z12, Z13 et Z14 dans sa fenêtre d'affichage, comme le présente la figure 18.

La zone Z11 est une zone qui n'est jamais couverte par la page P1 du plan supérieur. La zone Z11 contient donc en permanence la partie correspondante de la page P2 du plan inférieur.

La zone Z12 est la zone couverte par la page P1 du plan supérieur quand celle-ci n'est pas décalée, c'est-à-dire quand elle occupe la position illustrée sur la figure 14. Et lorsque la page P1 du plan supérieur est décalée (situation de la figure 15), la zone Z12 présente la partie correspondante de la page P2 du plan inférieur, cette page occupant alors la somme des zones Z11 et Z12.

La zone Z13 est la zone occupée par la pince, qui peut être symbolisée graphiquement de toute façon appropriée.

Enfin la zone Z14 est une zone qui est couverte par la page P1 du plan supérieur quelle que soit la position de cette dernière.

Nous allons maintenant présenter les différents états possibles du système et les transitions possibles entre ces états, ces dernières étant la conséquence des actions de l'utilisateur.

Les états sont définis par :

- la position de la page du plan supérieur (c'est-à-dire décalée ou non décalée)
- les positions possibles d'un pointeur de souris par rapport aux différentes zones,
- l'état de la pince PC (ouverte ou fermée).

La figure 19 présente les deux positions possibles de la page P1 du plan supérieur.

La figure 20 présente un exemple d'état du système, pour lequel est spécifié un ensemble de régions sensibles réagissant à l'arrivée dans celles-ci du pointeur de la souris, la position de la page du plan supérieur et l'état de la pince.

Dans l'exemple de la figure 20, les régions grisées (ici les zones Z12 et Z13) désignent des endroits où la présence du pointeur n'a pas d'effet sur le système, tandis que les régions blanches (ici les zones Z11 et Z14) désignent des régions pour lesquelles l'entrée du pointeur dans l'une de celles-ci provoque une action, comme on le détaillera plus loin.

Le comportement du système est présenté dans la figure 21 sous la forme d'un diagramme d'états-transitions, qui illustre les différentes manipulations des pages P1 et P2 dans leurs plans respectifs.

Le système fait en sorte que la page P1 du plan supérieur ne soit remplacée par la page P2 du plan inférieur (une fois que cette dernière a été chargée à partir d'un serveur, le cas échéant) que seulement sur action implicite de l'utilisateur, et ceci de manière très intuitive. Contrairement aux procédés de « streaming », l'utilisateur ne sera ainsi pas surpris de voir remplacée la page qu'il est justement en train d'examiner.

Ainsi, quand la page du plan inférieur est le cas échéant chargée :

- si le pointeur de la souris se trouve dans la zone Z11, le fait de l'amener dans la zone Z12
- si le pointeur se trouve dans l'une des zones Z12, Z13 et Z14, le fait de l'amener dans la zone Z11 puis de nouveau dans la zone Z12

aura comme conséquence de décaler la page P1 du plan supérieur.

Ensuite, l'utilisateur pourra provoquer le mouvement inverse de la page P1 du plan supérieur (c'est-à-dire la ramener dans la zone Z12) :

- soit en ramenant le pointeur de la souris dans la zone Z11,
- soit en ramenant le pointeur de la souris dans la zone Z14.

Ce double choix offert à l'utilisateur présente un avantage important :

- le premier choix permet d'utiliser les moyens d'interactivité (boutons, etc.) contenus dans la page P2 du plan inférieur dans la zone Z11,
- le deuxième choix permet d'utiliser les moyens d'interactivité contenus dans la page P1 du plan supérieur dans la zone Z12.

Le fait de « serrer » la pince PC (par exemple par un simple clic de souris dans la zone dédiée Z13) a pour effet d'éviter de décaler la page du plan supérieur à la suite des manipulations indiquées ci-dessus. En d'autres termes, le processus de décalage et de retour de la page P1 est alors neutralisé, et les positions mutuelles des pages P1 et P2 (que ce soit avec la page P1 décalée ou la page P1 non décalée) sont figées.

Cette neutralisation peut être supprimée avantageusement en cliquant à nouveau dans la zone Z13 pour « desserrer » la pince PC.

On notera que les différents états et les différentes transitions illustrés sur la figure 21 peuvent être complétés le cas échéant par d'autres états et d'autres transitions, notamment lorsqu'il existe dans la fenêtre, à l'extérieur des zones Z11 à Z14, une zone de marge qui peut être utilisée, par exemple, pour passer de la zone Z11 à l'une

des zones Z13 ou Z14, ou réciproquement, sans passer par la zone Z12 qui les sépare.

Le système décrit ci-dessus en référence aux figures 14 à 22 peut être étendu pour gérer plus de deux plans superposés.

Ainsi l'on a illustré sur les figures 23 à 26 une page P2 de plan inférieur qui est fixe, et deux pages P1 et P1' de niveaux respectivement supérieur et intermédiaire, qui peuvent individuellement adopter une position non décalée (pleinement visible) ou décalée. La pince PC est ici commune aux trois pages, et peut donc occuper une zone graphique unique.

En supplément de la pince, un moyen graphique de « solidarisation » des plans adjacents pris deux à deux peut être mis en œuvre. Quand ce moyen est activé, les deux plans qu'il relie se déplacent ensemble.

Le système décrit en référence aux figures 14 à 26 peut ici encore avantageusement être mis en œuvre dans le cadre d'applications sur l'Internet, dans lesquelles les pages mettent un temps significatif à être chargées. La page P1 du plan supérieur pourra être chargée en premier et faire patienter l'utilisateur pendant que la page P2 du plan inférieur se charge. De même, l'affichage des pages P1 et/ou P2 peut permettre de faire patienter l'utilisateur pendant le chargement d'une autre page P3, encore plus « inférieure » selon la présentation adoptée ici, et ainsi de suite.

Enfin, dans les différentes formes de réalisation de l'invention, on peut prévoir une page P1 sensiblement plus petite que la page P2, auquel cas les zones d'affichage occupées par la page P1 dans ses deux positions respectives sont disjointes.

On va maintenant décrire en référence aux figures 27 à 31un procédé de gestion de l'affichage de fenêtres d'un système d'exploitation notamment lors d'opérations de glisser-déposer.

La figure 27 illustre une première fenêtre F1 qui recouvre partiellement une seconde fenêtre F2 (peu importe ici que la fenêtre qui se trouve « sur le dessus » soit active ou non).

On a illustré sur la figure 28 un pointeur (ou curseur) PT de souris qui se trouve sur un objet O1 (tel qu'un dossier ou document) visible dans la fenêtre F1. L'appui par l'utilisateur d'un bouton de la souris permet, de façon connue en soi, de « saisir » cet objet en vue d'une opération, également connue en soi, de «glisser-déposer ».

Sur la figure 29, le pointeur P1 accompagné de l'objet O1 parcourt un trajet (flèche f1) de la fenêtre F1 vers la partie visible de la fenêtre F2, en franchissant donc le bord (gauche en l'occurrence) de la fenêtre F1, sans lâcher le bouton de la souris.

Sur la figure 30, le mouvement du pointeur PT accompagné de l'objet O1 atteint à nouveau, par un mouvement inverse (flèche f2), le bord de la fenêtre F1 préalablement franchi. Ce phénomène provoque, dans le présent exemple de réalisation, la mise en arrière-plan de la fenêtre F1 par rapport à la fenêtre F2.

L'entièreté de la fenêtre F2 devient donc visible, et l'utilisateur peut donc librement choisir lequel, parmi les objets visibles dans toute la fenêtre F2, celui des objets (objet désigné par O2) sur lequel il va déposer l'objet O1. Ceci est illustré par la figure 31.

Selon une variante, on peut prévoir que lors du mouvement du pointeur selon la flèche f2, la fenêtre F1 se déplace, de préférence selon le même mouvement que le pointeur PT, pour ainsi progressivement dégager la fenêtre F2.

D'autres variantes sont bien entendu possibles, et notamment :

- le système effectue une signalisation auprès de l'utilisateur lors de l'atteinte par le pointeur du bord de la première fenêtre ; cette signalisation peut être choisie dans le groupe comprenant une signalisation visuelle, une signalisation sonore et une signalisation tactile telle qu'un retour de force au sein d'un périphérique d'entrée manuel qui commande ledit pointeur ;
- la fenêtre F1 peut passer automatiquement en arrière-plan de la fenêtre F2 dès que le pointeur PT muni de l'objet O1 a séjourné dans la zone de la seconde fenêtre non recouverte par la première fenêtre pendant une durée supérieure à un seuil prédéterminé ;
- la fenêtre F1 peut passer automatiquement en arrière-plan de la fenêtre F2 lorsqu'une touche d'un clavier du poste informatique est actionnée, ou encore par commande vocale ;
- la mise en arrière-plan ou le déplacement de la fenêtre F1 pour dégager la fenêtre F2 peut être réversible, par exemple en effectuant un déplacement inverse du pointeur après son mouvement selon la flèche f2.

Bien entendu, l'homme du métier saura effectuer les adaptations nécessaires dans le cas où il faut gérer l'affichage de plusieurs fenêtres superposées.

* * *

Chapitre II - Création de liens (Figures 32 à 84)

Selon un autre aspect de l'invention, on va maintenant décrire un système permettant à l'utilisateur de créer des liens hypertexte (que l'on appellera dans la suite « liens ajoutés ») entre des pages qu'il découvre au gré de sa navigation sur l'Internet.

Un but de ce système est que chaque utilisateur ajoute ses propres liens dans la masse d'informations disponible sur l'Internet. La figure 32 illustre ce concept, chaque point noir représentant une page et chaque trait séparant deux points noirs représentant un lien ajouté.

Concrètement, l'utilisateur met en place de nouveaux liens au cours même de sa navigation, ces liens ajoutés étant destinés à l'aider dans sa navigation dans le futur. Il navigue ainsi de manière « pro-active ».

En d'autres termes, pour chaque utilisateur, l'ensemble des liens ajoutés, que l'on peut considérer comme sa toile miniature personnelle (Miniweb), s'ajoute à la toile Internet (Web) à laquelle il avait déjà accès, comme illustré sur la figure 33.

A cet effet, le système est installé entre le navigateur et les sites visités, et augmente pour chaque utilisateur le réseau des liens qui sont à sa disposition pour naviguer. Pour ce faire, le système est doté de moyens de stockage des liens ajoutés. Ces moyens permettent, à chaque accès (Link) à une page à laquelle un ou plusieurs liens ajoutés ont été associés, de présenter ces derniers à l'utilisateur en supplément de ladite page. Sur un plan pratique, et comme l'illustre la figure 34, une indirection est établie vers une table (Link→Added Links) de stockage des liens ajoutés, afin d'associer ces liens ajoutés (Added Links) à cette page avant de la présenter à l'utilisateur.

Le stockage des liens ajoutés peut être local (sur le poste client) ou distant (sur un serveur approprié). Dans la suite de la description, on se focalisera sur un stockage distant. En effet, dans certains des perfectionnements qui seront présentés plus loin, les liens formant les toiles miniatures personnelles de différents utilisateurs seront automatiquement comparés entre eux et des liens ajoutés pourront être communiqués d'une toile personnelle à une autre, ce qu'un stockage distant permet plus facilement de mettre en œuvre.

Chaque toile personnelle peut ainsi s'enrichir de liens ajoutés provenant d'autres toiles personnelles, et proposés automatiquement par le système qui par exemple possède en outre des moyens pour détecter que leurs utilisateurs respectifs partagent les mêmes centres d'intérêt. Par ailleurs, chaque utilisateur peut se constituer un

cercle de relations (d'amis) proposées par le système suite à la détection de centres d'intérêts communs.

La suite de la description est structurée de la manière suivante :

On présentera d'abord le système qui permet à un utilisateur de créer des liens ajoutés et de les retrouver ultérieurement (lors des accès aux pages auxquelles ils ont été associés). Ceci est décrit dans la section « Créer et retrouver des liens ajoutés ».

On présentera ensuite un perfectionnement du système consistant à combiner des liens ajoutés avec une approche de stockage de ces liens dans une structure hiérarchique de répertoires et sous-répertoires (c'est-à-dire en intégration avec l'approche classique des systèmes de mémorisation de « liens favoris », « signets », ou « bookmarks » ou encore « scrapbook » selon la terminologie anglo-saxonne). Ceci sera décrit dans la section « Création de liens ajoutés dans des Répertoires ».

On présentera ensuite un perfectionnement du système classique de marque-pages, qui consiste à rendre les marque-pages contextuels.

On présentera ensuite un perfectionnement du système permettant aux utilisateurs de publier (ou mettre à disposition d'un groupe restreint) au moins un sous-ensemble des liens ajoutés de leur toile personnelle. Ces liens ajoutés pourront être visualisés directement au moyen d'un navigateur Internet standard. Alternativement, ces liens ajoutés pourront être consultés avec le système et enrichir la toile personnelle de l'utilisateur qui les consulte. Ceci est décrit dans la section « Publication de liens ajoutés ».

On présentera ensuite un perfectionnement qui permet au système de suggérer automatiquement aux utilisateurs des liens ajoutés provenant d'autres utilisateurs dont les centres d'intérêt sont proches. Ceci est décrit dans la section « Détection de Répertoires Proches ».

On présentera ensuite une architecture qui permet à l'éditeur ou l'administrateur d'un site Internet (« Webmaster » en terminologie anglo-saxonne) de proposer des liens ajoutés aux utilisateurs du système de manière personnalisée. Ceci est décrit dans la section « Suggestion de liens ajoutés par le Webmaster ».

On présentera enfin un système permettant de simuler la propagation d'un nouveau lien ajouté parmi les utilisateurs du système sur l'Internet et de calculer le volume de l'audience probable. Ceci est décrit dans la section « Calcul de l'Audience ».

On présentera ensuite un perfectionnement du système permettant à l'utilisateur de publier (ou mettre à disposition d'un groupe restreint) au moins un sous-ensemble des répertoires de sa toile personnelle. Ces répertoires peuvent être visualisés directement au moyen d'un navigateur standard. Alternativement, ces répertoires peuvent être consultés avec le système et enrichir la toile personnelle de l'utilisateur qui les consulte. Ceci est décrit dans la section « Publication de Répertoires ».

On présentera ensuite un perfectionnement du système permettant à l'utilisateur de stocker (en local ou sur un serveur) non seulement un lien mais aussi une copie figée d'une page. Ceci est décrit dans la section « Liens en mode Gelé ».

On décrira ensuite un perfectionnement qui, dans le cadre du système présenté jusque là, permet de publier des pages qui ont été restructurées pour présenter des liens directement à l'intérieur d'elles-mêmes, l'utilisateur pouvant supprimer ces liens ou en ajouter d'autres. Pour ce faire, les pages contiennent un ou plusieurs contenants dans lesquels sont présentés des contenus qui peuvent eux-mêmes contenir d'autres contenants récursivement. Ceci est décrit dans la section « Contenants et contenus ».

En intégrant les concepts présentés jusque là (liens ajoutés associés aux pages et liens dans des contenants inclus dans des pages) et en les illustrant au moyen d'un exemple, on récapitulera ensuite les fonctionnalités du système global et on les

complétera par quelques perfectionnements. Ceci est décrit dans la section « Proposition de Répertoires Spécialistes et Voisins».

On présentera ensuite un perfectionnement du système permettant à l'utilisateur d'associer aux liens qui l'ont intéressé des attributs, tels que le souhait d'achat ou de vente du produit représenté par l'information pointée (ou plus généralement des attributs de type « Offreur » et « Demandeur »). Ces attributs sont exploités par le système pour affiner la sélection de répertoires spécialistes et proches, et la mise en relation d'utilisateurs intéréssés par de mêmes contenus et se trouvant dans une situation de complémentarité. Ceci est décrit dans la section « Attributs relatifs à l'acceptation ».

On présentera ensuite un perfectionnement qui permet aux pages restructurées de présenter des contenus de manière « personnalisée », c'est à dire en fonction des centres d'intérêts de l'utilisateur. Ceci est décrit dans la section « Pages Personnalisées ».

On décrira ensuite un ensemble de perfectionnements du système, permettant à un utilisateur de charger une page progressivement, et à des utilisateurs distants de partager le contenu de pages qu'ils manipulent simultanément dans le cadre d'applications collaboratives sur un réseau tel que l'Internet.

Section 1 - Créer et retrouver des liens ajoutés

La Figure 35 illustre le principe selon lequel, en conséquence d'un accès par l'utilisateur à une page p1 via l'Internet, le système lui présente automatiquement un certain nombre de liens ajoutés, en l'occurrence sous forme de symboles graphiques, vers des pages p2, p3 et p4 qu'il avait associées à la page p1 au cours de navigations précédentes.

Motivation

Les liens ajoutés sont créés par l'utilisateur dans le but d'établir des relations entre des pages qui l'intéressent. Les liens ajoutés représentent des relations pertinentes entre ces pages ; une page liée à une autre page est intéressante dans le contexte de ladite autre page (et optionnellement, dans le sens inverse également). Pour l'utilisateur, l'ensemble des liens ajoutés constitue donc un *réseau* de relations supplémentaires entre pages qui l'intéressent les unes par rapport aux autres, comme l'illustrent les figures 36a, 36b et 36c qui illustrent pour les figures 36a et 36b, deux réseaux de liens entre un certain nombre de pages, et pour la figure 36c, le réseau « aggloméré » desdits liens.

A partir d'une page ou d'un ensemble de pages sélectionné, le système permet à l'utilisateur de naviguer vers des pages qui y sont reliées. Le réseau de liens ajoutés constitue ainsi pour l'utilisateur un réseau de navigation personnel selon son propre « schéma mental » et lui permet de retrouver plus facilement les éléments d'information qu'il avait déjà repérés.

Créer des liens ajoutés

L'approche des plans coulissants décrite dans la première partie de la présente description est particulièrement favorable à l'établissement de liens ajoutés par la technique, classique en soi, du « glisser-déposer ». L'utilisateur peut sélectionner une page ou un élément (ou zone graphique qui la représente) qui se trouve dans un plan, la glisser et la déposer dans une autre page (ou zone graphique qui la représente) qui se trouve dans un autre plan, afin d'établir un lien ajouté entre ces deux pages.

Cette approche est à comparer avec l'approche qui consisterait à glisser-déposer des pages entre des fenêtres différentes de l'outil de navigation, qui se recouvrent partiellement et qui empêchent donc de visualiser les pages entièrement. L'avantage de l'approche par plans coulissants est de permettre, *au cours même du processus de glisser-déposer* (c'est-à-dire sans lâcher le doigt sur le bouton de la souris), de visualiser le contenu des plans (les pages et leurs liens ajoutés) à tour de rôle, par

simple mouvement de la souris, comme décrit précédemment en référence notamment à la figure 26.

On comprend également que, selon une variante, cette technique de « glisser-déposer » avec révélation des zones d'affichage en second plan sur lesquelles se dirige le curseur au cours même de l'opération peut être généralisée au cas où ces zones d'affichage sont des fenêtres d'un systèmes d'exploitation classique tel que « Windows » ou « MacOS » (marques déposées). Plus précisément, on prévoit dans ce cas qu'une fenêtre située en second plan passe en premier plan quand le curseur « pousse » la fenêtre qui était en premier plan en arrivant contre l'un de ses bords.

La figure 37 présente un exemple, de création de liens ajoutés entre pages, en tirant des liens entre les objets graphiques les représentant (situées à gauche des pages), la pince décrite plus haut étant active et bloquant le mouvement des plans.

Le bas de la figure 37 illustre le cas particulier de création de liens ajoutés (par glisser-déposer d'objets graphiques) entre différentes instances du système, ouvertes simultanément par l'utilisateur sur le poste client. Les mouvements de souris qui permettent de tirer des liens ajoutés entre les pages P6 et P7 illustrent la technique décrite précédemment qui consiste à amener le pointeur de la souris à l'extérieur de la fenêtre contenant la page P6 puis le faire revenir en direction de cette même fenêtre afin de la faire passer en arrière-plan de la fenêtre de la page P7.

Bien évidemment, chaque plan peut être supprimé ou « minimisé » (de manière à ce que sa partie visible prenne le moins de place possible) au moyen de clic sur un bouton (ceci n'est pas illustré dans les figures).

Bien que la figure 37 ne le montre pas, des objets graphiques à l'intérieur des pages peuvent aussi être glissés-déposés, soit entre eux, soit vers des zones ou objets graphiques représentant les pages. Dans certains cas, comme on le verra plus loin, les

objets graphiques situés à l'extérieur des pages, et représentant des contenus de pages, peuvent aussi être glissés-déposés à l'intérieur de pages.

La figure 38 représente la disposition des plans coulissants quand le plan supérieur est décalé vers la droite. La figure 39 représente la disposition des plans coulissants quand tous les plans, sauf le plan le plus inférieur, sont décalés. Comme déjà décrit, la transition (d'une des dispositions de plans coulissants à une autre) se fait de façon incrémentale, au moyen de simple mouvements de la souris, la pince décrite plus haut étant débloquée.

On notera ici que, pour simplifier les manipulations, l'état de la pince peut être inversé par exemple en pressant sur la barre d'espace ou une autre touche du clavier du poste client.

Selon une variante, non représentée, la pince peut être déplacée sur un « rail » et il ne bloque que les plans situés à sa gauche. En d'autres termes, selon la position de la pince, les plans qui ne sont pas visibles à gauche de la pince sont bloqués et ceux qui sont à sa droite peuvent être décalés.

Selon une autre variante, on peut prévoir des pinces sélectivement activées et désactivées entre deux pages adjacentes, de manière à ce que ces deux pages soient toujours, dans le cadre du processus de déplacement de plans, traitées de la même façon.

On notera également ici que c'est le fait que la pince soit à l'état actif qui permet de tirer des liens ajoutés entre les pages (par glisser-déposer d'objets graphiques les représentant), sans entraîner le déplacement des plans tel que décrit.

Selon une autre forme de réalisation, les zones graphiques qui représentent des pages peuvent se trouver sur d'autres plans que ceux qui présentent les pages. La figure 40

présente ainsi une configuration où les liens ajoutés de chaque page sont sur un ou plusieurs plans adjacents à ladite page.

Les liens ajoutés peuvent aussi avantageusement être vues sous forme de vignettes comme des papillons « collés » sur les pages et déplaçables par l'utilisateur à souhait à l'aide de la souris.

Selon une variante avantageuse, on peut disposer ces vignettes dans des zones vierges de la page qui auraient pu être utilisées par exemple pour afficher des bandeaux publicitaires.

Les liens ajoutés par l'utilisateur par glisser-déposer (ou le cas échéant manuellement à l'aide du clavier ou par des clics de souris), sont stockés dans le système par exemple sous forme d'une table TABLE, comme l'illustre la figure 41.

Dans la suite, nous adoptons l'hypothèse (optionnelle) selon laquelle les liens ajoutés sur une page entraînent la création de liens inverses sur les pages pointées par ces liens ajoutés (bidirectionalité des liens). Ainsi, comme l'illustre la figure 42, en conséquence de la création des liens ajoutés de p2,p3 et p4 sur la page p1, un lien ajouté p1 s'ajoute automatiquement sur les pages p2, p3 et p4.

La création de liens ajoutés s'effectue quand l'utilisateur repère une relation entre différentes pages qu'il découvre au gré de sa navigation en suivant des liens hypertexte ou en se déplaçant à l'intérieur de grandes pages ou de mondes virtuels en trois dimensions (en réalité virtuelle immersive). Dans tous les cas, la création d'un lien ajouté peut se faire par simple glisser-déposer de la « poignée » ou autre miniaturisation de la page ou la scène couramment visualisée (par exemple de la scène 3D couramment visualisée) vers la poignée de la page à relier (autre scène 3D).

Les moyens de stockage des liens ajoutés sont avantageusement mis en œuvre de manière distante (sur un serveur), comme l'illustre la figure 43. L'utilisateur ajoute

et/ou supprime des liens ajoutés sur le poste client et ces manipulations sont répercutées sur le serveur, les communications se faisant par exemple par l'intermédiaire d'un Fournisseur d'Accès à l'Internet.

Section 2 - Retrouver les liens ajoutés

L'affichage de la page avec ses liens ajoutés peut être préparé sur le poste client, dans une architecture schématisée par le diagramme de la figure 44, ou sur le poste serveur, dans une architecture schématisée par le diagramme de la figure 45. On notera que les deux architectures sont adaptées à un fonctionnement selon le protocole HTTP par exemple.

Dans le cas de l'architecture de la figure 44, la requête ou lien « Link » de l'utilisateur est communiquée par le poste client, tant au serveur de la page demandée qu'au serveur où ont été stockés les liens ajoutés. En retour, avant présentation à l'utilisateur, les liens ajoutés fournis par le deuxième serveur sont graphiquement combinés, par le poste client, avec la page fournie par le premier serveur, et sont présentés conjointement à cette page.

Dans le cas de l'architecture de la figure 45, la requête ou lien de l'utilisateur est communiquée par le poste client, uniquement au serveur où ont été stockés les liens ajoutés. Ce serveur se charge de retransmettre la requête au serveur de la page demandée pour recevoir cette page. En retour, les liens ajoutés sont ajoutés à la page et le tout constitue la réponse à la requête de l'utilisateur. La combinaison graphique des liens ajoutés avec la page peut être effectuée sur le serveur ou sur le poste client.

Section 3 - Création de liens ajoutés dans des Répertoires

On connaît déjà dans l'état de la technique le concept dit de « marque-page » ou « signet » qui permet à un individu de mémoriser des liens vers ses pages favorites sur la Toile, un tel concept étant présent dans les navigateurs Internet courants. Ces

derniers permettent de ranger les liens mémorisés dans des répertoires qui peuvent former une structure arborescente de répertoires et de sous-répertoires.

L'établissement d'une toile personnelle (réseau personnel de navigation qui s'ajoute à la Toile) en créant (par glisser-déposer ou manuellement) des liens ajoutés directement sur des pages comme on l'a décrit dans la section précédente, est complémentaire à ces moyens classiques de rangement de liens dans des répertoires (constituant généralement des catégories personnelles de l'internaute).
En effet :

- l'utilisateur peut ranger dans un répertoire non seulement un lien (vers une page), mais aussi un lien ajouté (entre deux pages).

- les liens ajoutés peuvent ainsi être contextuels : un lien ajouté associé à une première page (ou plus exactement, associé à un lien vers une première page) et pointant sur une deuxième page peut exister dans le cadre d'un répertoire donné qui contient un lien vers cette première page sans exister dans un autre répertoire qui pourtant contient aussi le lien vers ladite la première page.

Par exemple, dans le répertoire « mes boissons favorites » sur la page « coca » (marque déposée) peut se trouver un lien ajouté vers la page « pepsi » (marque déposée) alors que dans le répertoire « mes styles de management préférés » sur la même page « coca » peut ne pas se trouver de lien ajouté vers la page « pepsi ». (Voir l'exemple à la fin de cette section et la figure 51).

(Toutes les marques citées dans le présent mémoire sont des Marques Déposées de leurs titulaires respectifs, notamment « coca », « pepsi », « perrier » et « hp »).

Dans les choix de conception adoptés dans toute la suite de la description :

- chaque lien ajouté appartient à un répertoire et un seul : un « répertoire courant » existe toujours, et c'est dans ce répertoire que les liens vers des pages et leurs liens ajoutés sont insérés par défaut ; si aucun répertoire n'est sélectionné par l'utilisateur pour être le répertoire courant, un premier répertoire courant lui est fourni par défaut quand il accède à une page et lui associe un premier lien ajouté.

- la création d'un lien ajouté dans un répertoire donné engendre la création du lien ajouté inverse dans le même répertoire (s'il n'y existe pas déjà). Cette règle impose l'existence, dans le même répertoire, du lien vers la page pointée par le lien ajouté ; dans le cas où il n'existe pas, il y est automatiquement inséré ; ceci est illustré par l'exemple de la figure 46).

- l'utilisateur peut sélectionner plusieurs répertoires comme étant « répertoires courants » ; les liens ajoutés visualisés pour une page courante sont alors constitués par l'union des liens ajoutés situés dans ces répertoires ; un nouveau lien ajouté inséré l'est dans chacun de ces répertoires ; enfin on verra dans la section « Détection de Répertoires Proches » que la recherche de répertoires proches se fera également sur la base de l'union des liens ajoutés situés dans ces répertoires.

L'utilisateur peut effectuer les actions suivantes :

- insérer ou supprimer, dans un répertoire, un lien vers une page :

* pour insérer une page, une des méthodes possibles est la suivante : l'utilisateur choisit un répertoire courant (ou plusieurs, voir l'explication du paragraphe précédent), ou reste dans le répertoire par défaut (qui est considéré comme répertoire courant), et accède à une page dont le lien sera automatiquement rangé dans ce(s) répertoire(s) courant(s).

* pour supprimer une page, l'utilisateur sélectionne la page en question et actionne la commande supprimer (par exemple par la touche « Suppr » du clavier).

- créer un lien ajouté :

* soit il crée manuellement un lien ajouté sur une page (typiquement en tapant son adresse URL au clavier),

* soit il utilise la technique du glisser-déposer : à partir d'une première page, il tire dans une deuxième page un lien ajouté qui pointera vers la première page.

- déplacer ou copier une ou plusieurs pages d'un répertoire à un autre :

* soit, manuellement, il choisit les pages à déplacer ou copier, soit il utilise la technique de glisser-déposer ; on notera qu'en fait, l'utilisateur déplace ou copie des liens vers des pages, même s'il croit déplacer les pages elles-mêmes.

Dans la suite de la description, pour simplifier la lecture et décrire ce que l'utilisateur « croit faire », nous utiliserons parfois le mot « page » au lieu de « lien vers une page ». Ainsi l'on dira « déplacer une page » (ou « copier une page ») en lieu et place de « déplacer un lien vers une page » (« copier un lien vers une page »). On dira aussi « une page possède un lien ajouté » au lieu de dire « à un lien vers une page est associé un lien ajouté », ou encore « lien ajouté sur une page » au lieu de « lien ajouté associé à un lien vers une page ». De plus, « lien vers une page » est un raccourci pour « lien vers une page ou vers un élément contenu dans une page ».

Prenons un exemple:

L'utilisateur a créé deux répertoires, libellés respectivement « Drinks » (pour « mes boissons favorites ») et « Mgt » (pour « styles de management »). Il a aussi à sa disposition un répertoire par défaut, libellé « Buffer » (pour « tampon ») qui est créé automatiquement par le système.

Selon l'approche classique de mémorisation des marque-pages, l'utilisateur insère le lien www.coca.com (« coca ») dans le répertoire Drinks. Il insère aussi le lien www.hp.com (« hp ») dans le répertoire Mgt.

Dans le cadre du répertoire courant qui en l'occurrence est Buffer (par défaut), l'utilisateur accède via la Toile à la page www.pepsi.com (« pepsi »). A partir de coca (qui se trouve dans Drinks) il tire (par glisser-déposer) un lien ajouté sur la page pepsi (voir figure 46).

Ce lien ajouté sur la page Pepsi va être www.coca.com. Comme, pour satisfaire l'hypothèse de conception mentionnée précédemment, un lien ajouté inverse doit être créé dans le même répertoire, la page coca doit y être introduite avec un lien ajouté www.pepsi.com.

En résultat, les pages pepsi et coca se trouvent tous les deux dans Buffer, chacun ayant un lien ajouté sur l'autre. En effet, le lien sur la page coca a dû être dupliqué dans Buffer pour permettre la création du lien ajouté inverse (de coca vers pepsi). La figure 46 illustre cet exemple.

L'interface homme-machine permettant ces opérations peut par exemple être celle schématisée dans les figures 46 à 51.

L'utilisateur peut, par glisser-déposer, copier ou déplacer une page vers un autre répertoire. L'utilisateur peut aussi copier ou déplacer un ensemble de pages d'un répertoire à un autre. Plusieurs cas de commande au système sont possibles :

1. soit ceci entraîne le déplacement ou la copie des liens vers ces pages, sans leurs liens ajoutés. (Voir figure 47) ;

2. soit ceci entraîne le déplacement ou la copie des liens vers ces pages, avec tous les liens ajoutés existant entre elles ; le déplacement n'est possible que pour les pages

qui n'ont pas de liens ajoutés avec des pages externes à l'ensemble déplacé (les autres pages de l'ensemble sont copiés) ; voir figure 47 ;

3. soit ceci entraîne le déplacement ou la copie des liens vers ces pages, avec tous les liens ajoutés entre elles ou avec d'autres pages du même répertoire ; voir figure 48 ;

En poursuivant le même exemple, l'utilisateur déplace la page coca du répertoire Buffer vers le répertoire Drinks ; si l'on se trouve dans le premier ou deuxième cas de commande au système, la page coca se trouvant déjà dans Drinks, cette opération n'a aucun effet. Ceci est illustré par la figure 47.

Dans un autre cas de figure, l'utilisateur voudrait ranger dans le répertoire Drinks tous les liens ajoutés de la page coca. Si l'on se trouve dans le troisième cas de commande au système, en résultat du glisser-déposer de coca (qui se trouve dans Buffer) vers le répertoire Drinks, la page pepsi se retrouve également dans Drinks et le lien ajouté entre pepsi et coca est reproduit dans Drinks. L'utilisateur pourrait ensuite supprimer du répertoire Buffer les pages pepsi et coca (avec tous les liens ajoutés entre elles) ; ces deux opérations sont illustrées sur la figure 48.

On notera que pepsi (ou coca) n'aurait pas pu être supprimé si une troisième page « perrier » (marque déposée) se trouvait dans le même répertoire Buffer et avait un lien ajouté sur pepsi ou sur coca), à moins que cette troisième page « perrier » ne fut supprimée dans la même action. Ceci est illustré par la figure 49.

Si l'on se trouve dans le deuxième cas de commande au système, l'utilisateur peut directement déplacer l'ensemble des liens pepsi et coca vers le répertoire Drinks, ainsi que les liens ajoutés entre eux. Aucune page externe à l'ensemble supprimé n'ayant un lien ajouté vers coca, cette page est alors supprimé d'office. Ceci est illustré sur la figure 50.

Poursuivons l'exemple : l'utilisateur, qui pense que les styles de management des sociétés coca et hp (marques déposées) méritent d'être mis en relation, tire par glisser-déposer un lien ajouté sur la page hp à partir de la page coca. Ceci est illustré à la figure 51.

La création du lien ajouté de hp vers coca entraîne la création du lien ajouté inverse de coca vers hp, les deux liens ajoutés se trouvant dans le répertoire Mgt (où se trouve la destination du glisser-déposer). Bien que l'opération effectuée ne consiste pas explicitement en un déplacement ou une copie de page, elle implique la copie de la page coca dans le répertoire Mgt (puisque les deux extrémités de tout lien ajouté doivent se trouver dans un même répertoire) .

Section 4 - Marque-pages contextuels

Rappelons que le système classique de marque-pages (déjà mentionné au début de la section précédente) permet à l'utilisateur de ranger dans un répertoire un lien sur (ou une copie de) la page courante visualisée, dans le cas où il veut mémoriser ledit lien.

De plus, les navigateurs modernes offrent l'option d'afficher, dans une sous-fenêtre adjacente à celle contenant la page visualisée, la structure arborescente de répertoires de marque-pages et les marque-pages (liens ou copies mémorisés) qu'ils contiennent.

Cette option d'affichage des liens mémorisés, simultanément à la page accédée par l'utilisateur, peut être perfectionnée en rendant contextuel l'affichage des répertoires de liens mémorisés et des liens (ou copies) mémorisés eux-mêmes. En effet, plutôt que de présenter la structure des répertoires et les liens qu'ils contiennent, de manière indépendante de la navigation de l'utilisateur, le système peut directement :

- pointer sur les répertoires contenant déjà le lien sur la page courante visualisée,
- et présenter leurs contenus respectifs.

Ainsi, l'utilisateur n'a pas à aller chercher, dans ses répertoires, les liens (sur les pages qui l'avaient intéressé) mémorisés qui se trouvent être dans le contexte de la page courante, puisqu'il les a automatiquement à sa disposition.

Dans une mise en œuvre pratique, le système présente ces répertoires (contenant la page courante) dans la zone présentant les liens ajoutés associés à la page courante en leur donnant une apparence différente (l'icône représentant un répertoire ou une page d'un répertoire qui n'est pas associé par un lien ajouté à la page courante est différente de l'icône représentant une page associée par lien ajouté). Ceci est schématisé dans la figure 52 qui illustre l'affichage des répertoires dans lesquels est rangé la page courante, automatiquement, simultanément à l'affichage de la page courante. L'utilisateur peut, par un simple clic sur un bouton (par exemple sur la barre comportant un triangle sur la figure 52), visualiser le contenu de ces répertoires et retrouver ainsi les autres pages qu'il avait rangées ensemble dans ces répertoires.

Selon un détail de mise en œuvre, dans le cas où la page en question n'est rangée que dans un seul répertoire, les autres pages dudit répertoire (si elles ne sont pas trop nombreuses) peuvent être affichées directement, sans même nécessiter un clic, pour offrir ainsi un accès contextuel aux liens mémorisés de manière encore plus directe.

L'utilisateur pourra cliquer directement sur la présentation graphique représentant un répertoire, par exemple une icône représentant un dossier (r2 sur la figure 52) pour ouvrir une nouvelle page présentant le répertoire r2 avec tout son contenu (les sous-répertoires et les pages qu'il contient).

Bien évidemment, la présentation de la figure 52 peut être remplacée par une présentation classique d'arbre comme celle que l'on trouve dans les systèmes de marque-pages existants.

Une fonction analogue concerne l'affichage des liens Ajoutés associés à une page courante. Il s'agit de pouvoir naviguer au moyen de liens Ajoutés « de deuxième rang ».

La figure 53 illustre un exemple de mise en œuvre de cette approche. En cliquant sur une zone graphique particulière d'un lien Ajouté p3 associé à la page p1, l'utilisateur peut voir également les liens Ajoutés (p31, p32 et p33) qui sont associés à la page p3 alors que c'est la page p1 qui est couramment visualisée (et non la page p3).

Bien évidemment, cette approche peut être généralisée (récursivement) pour l'affichage de liens Affichés de rangs supérieurs à 2.

On observera en outre que chacune des présentations graphiques représentant un lien mémorisé ou un lien ajouté (par exemple une vignette représentant la page sous forme miniaturisée) peut être accessoirement dotée d'un signe qui indique que la page pointée a été mise à jour, le cas échéant, depuis la dernière visite de l'utilisateur. Ceci constitue une invitation à aller consulter à nouveau ladite page.

En outre, ce même signe peut figurer sur la représentation graphique de chaque répertoire dont au moins l'une des pages a été mise à jour.

Section 5 - Publication de liens ajoutés

Un perfectionnement du système consiste à permettre de publier les liens ajoutés que l'on a associés à une page. Par publier, on entend le fait de les mettre à la disposition du public ou d'un groupe déterminé d'utilisateurs (qui pourront y accéder par exemple en donnant un mot de passe).

Le système permet deux cas d'accès aux liens ajoutés publiés :

- les liens ajoutés publiés peuvent être visualisés directement au moyen d'un navigateur Internet standard,

- ou les liens ajoutés publiés être consultés au moyen de l'interface cliente du système et enrichir la toile personnelle de l'utilisateur qui les consulte.

Visualisation directe au moyen d'un navigateur standard

Dans le cas d'une visualisation directe au moyen d'un navigateur standard, un premier utilisateur visualise par des moyens standard une page publiée par un deuxième utilisateur sur la Toile, ladite page possédant des liens ajoutés associés à une autre page. Cette visualisation peut être le fruit d'un accès ou d'une réception :

- le premier utilisateur peut accéder via la Toile à une page présentant les liens ajoutés publiée par le deuxième utilisateur, de manière classique, en utilisant son navigateur Internet standard et en communiquant l'adresse URL de la page,

- le premier utilisateur peut recevoir une page présentant les liens ajoutés, publiée par le deuxième utilisateur, de manière classique par courrier électronique.

La page à laquelle il a été accédé via la Toile ou reçue par courrier électronique comporte des champs permettant au premier utilisateur de s'identifier pour le cas où il voudrait consulter les liens ajoutés au moyen du système. Il s'agit donc d'une « invitation » à utiliser le système, dont les avantages principaux sont :

- de permettre de mémoriser les liens ajoutés dans son espace personnel sur un serveur et de les retrouver automatiquement , et
- de recevoir automatiquement des suggestions de nouveaux liens ajoutés à partir de tiers.

Un exemple schématique de cette « invitation » en est illustré à la figure 54.

Consultation au moyen du système

Dans le cas d'une consultation au moyen du système, lorsqu'un premier utilisateur accède, pour une page donnée, aux liens ajoutés chez un deuxième utilisateur, ces liens ajoutés s'ajoutent à ceux qui, le cas échéant, avaient déjà été associés à cette même page chez le premier utilisateur.

On va maintenant décrire la manière dont un deuxième utilisateur est sélectionné par le premier utilisateur pour y puiser des liens ajoutés.

Pour chaque page à laquelle accède le premier utilisateur, trois listes de répertoires appartenant à des deuxièmes utilisateurs lui sont proposées, comme l'illustre la figure 55a. L'utilisateur peut sélectionner (par exemple en cliquant une case à cocher) un ou plusieurs éléments de ces listes pour indiquer qu'il souhaite ajouter leurs liens ajoutés dans sa propre liste de liens ajoutés. Les trois listes sont décrites ci-dessous :

1. le contenu de la première liste est optionnel : ce contenu existe dans le cas où l'auteur de la page en question est lui-même un utilisateur du système et a voulu suggérer des liens ajoutés au premier utilisateur ; ceci est décrit en détail plus loin dans la section « Suggestion de liens ajoutés par l'administrateur du site » ;

2. la deuxième liste (« voisins d'intérêt ») a pour objet de présenter au premier utilisateur les répertoires ayant le plus de probabilité de contenir des liens ajoutés intéressants par rapport à son profil d'intérêts ; la liste de ces répertoires est déterminée automatiquement par le système en comparant les toiles personnelles des utilisateurs ; ceci est décrit plus loin dans la section « Détection de Répertoires Proches »).

3. la troisième liste (« copains ») contient des répertoires choisis par le premier utilisateur (par exemple, des répertoires appartenant aux toiles personnelles d'amis, collègues, experts, etc.).

Selon une variante, peut s'ajouter aux trois listes ci-dessus une quatrième liste des répertoires appartenant au premier utilisateur.

Dans le premier cas présenté plus haut (visualisation directe au moyen d'un navigateur standard), le pseudonyme du deuxième utilisateur qui est l'émetteur des liens ajoutés reçus par le premier utilisateur – ou du répertoire contenant ces liens ajoutés (on notera ici qu'il peut également s'agir d'un autre répertoire du premier utilisateur) – est présenté d'office en première position dans la troisième liste (« copains ») en mode actif (c'est-à-dire sélectionné) par défaut.

La troisième liste, peut avantageusement être basée sur :
- une « liste par défaut » donnée globalement par l'utilisateur pour toutes les pages ;
- et « surchargée » par les préférences fournies par l'utilisateur expressément pour la page en question.

Dans l'ensemble de liens ajoutés associés à la page qu'il est en train de visualiser, l'utilisateur reçoit les liens ajoutés supplémentaires puisés dans les répertoires sélectionnés dans ces trois listes. On peut ainsi considérer que l'utilisateur « visite » ces répertoires.

Des moyens pour sélectionner des liens ajoutés, sur la base de critères données par l'utilisateur ou associés à la page courante, peuvent être avantageusement mis en œuvre. Par exemple, l'utilisateur peut spécifier qu'il ne veut puiser dans les répertoires cochés que des liens ajoutés sur des pages étiquettées :
- « Demandeur » plutôt que « Offreur », afin de découvrir de nouveaux clients plutôt que de nouveaux vendeurs,

- selon des critères concernant l'audience visée : « Age », « Zone Géographique », « Sexe », « Préférence » (parmi un ensemble fini de préférences),
- etc.

Les liens ajoutés qui lui sont ainsi suggérés doivent être distingués de ceux qu'il avait lui-même ajoutés, de manière à ce qu'il puisse sélectivement les accepter et les refuser. Dans ce but, les liens ajoutés peuvent se trouver dans un parmi plusieurs modes différents: « Suggéré », « Accepté », « Refus » et « Gelé » (ce dernier mode étant décrit plus loin dans la section « Liens en mode Gelé »).

On donnera plus loin dans le chapitre III une description plus complète de la mise en œuvre technique de ces modes.

Les liens ajoutés reçus par un premier utilisateur, à partir des liens ajoutés publiés par un deuxième utilisateur (ou à partir des liens ajoutés d'un autre répertoire du premier utilisateur), sont au départ chez le premier utilisateur en mode « Suggéré ».

Suite à une action de l'utilisateur, chaque lien ajouté en mode Suggéré peut passer à l'un parmi les autres modes. L'interface homme-machine permettant ceci est schématisée à la figure 55b.

A cet effet, les moyens de stockage associent aux liens ajoutés un attribut « Mode » qui peut prendre comme valeur les modes déjà mentionnés à l'exception de Suggéré (c'est-à-dire, l'une parmi trois valeurs : Accepté, Refusé, Gelé). La figure 56 présente un diagramme de classe pour les liens ajoutés, selon la représentation standard UML (« Unified Modeling Language ») mettant en évidence l'attribut Mode des liens ajoutés.

Un lien ajouté en mode Suggéré chez un premier utilisateur n'existe que pendant la « visite » des liens ajoutés (pour la page en question) chez un deuxième utilisateur (ou dans un autre répertoire du premier utilisateur). Il n'est pas affiché en dehors de

cette visite. Rappelons que cette visite se fait dès que et tant que un ou plusieurs répertoire(s) est (sont) sélectionné(s) dans l'une des listes (Spécialistes, Voisins, Copains).

Un lien ajouté en mode Suggéré n'est pas stocké dans l'espace de stockage personnel du premier utilisateur car, à chaque nouvelle visite, il est recréé à partir du lien ajouté correspondant stocké chez le deuxième utilisateur, ou encore dans l'autre répertoire du premier utilisateur.

Le passage d'un lien ajouté du mode Suggéré au mode Refusé a pour effet de filtrer automatiquement ledit lien ajouté lors des accès ultérieurs à l'ensemble des liens ajoutés stockés dans l'espace mémoire du deuxième utilisateur, ou dans un autre répertoire du premier utilisateur, pour la même page.

Le passage au mode Accepté signifie que l'utilisateur valide le lien ajouté qui est en mode Suggéré. Cette action a pour effet :

1. de stocker ledit lien ajouté dans son espace de stockage personnel, de manière à ce qu'il lui soit présenté à nouveau avec chaque nouvelle présentation de la page à laquelle ledit lien ajouté est associé ; cette présentation a lieu même si ledit lien ajouté a été supprimé à la source (chez le deuxième utilisateur, ou dans l'autre répertoire du premier utilisateur) – ceci par opposition aux liens ajoutés en mode Suggéré qui ne sont plus présentés dès le moment où ils sont supprimés chez le deuxième utilisateur ; en outre, cette présentation a lieu même si la source (à savoir le deuxième utilisateur ou l'autre répertoire du premier utilisateur) n'est plus sélectionnée (a été décochée).

2. si le répertoire dans lequel ledit lien ajouté est rangé, est publié, de publier d'office (sauf instruction contraire de la part de l'utilisateur) ledit lien ajouté – ceci par opposition aux liens ajoutés en mode Suggéré qui eux ne seront pas publiés.

L'insertion (manuelle ou par glisser-déposer) d'un lien ajouté sur une page entraîne d'office sa mise en mode Accepté (puisque l'utilisateur l'a inséré lui-même).

Section 6 - Détection de Répertoires Proches

Présentation générale

Le système détecte pour l'utilisateur l'existence d'autres utilisateurs qui partagent ses intérêts (et ses goûts) par rapport à une « page courante » qu'il est en train de visualiser et qui représente donc le contexte courant de ses intérêts.

Ainsi, par rapport à une page courante visualisée par un premier utilisateur, le système sélectionne les deuxièmes utilisateurs les plus « proches », c'est-à-dire ayant, dans leur réseau de liens ajoutés publiés, d'une part ladite page, et d'autre part, autour de ladite page, l'ensemble le plus grand de liens ajoutés en commun avec le premier utilisateur.

Pour chacun des deuxièmes utilisateurs sélectionnés par le système, les liens ajoutés associés à ladite page qui ne figurent pas chez le premier utilisateur sont « suggérés » à ce dernier.

En variante, la sélection des deuxièmes utilisateurs proches peut se faire par rapport à un ensemble de pages sélectionnées par le premier utilisateur, plutôt que par rapport à une seule page. On va décrire dans la suite le procédé de sélection à partir d'une seule page, l'homme de métier saura adapter cette méthode au cas de plusieurs pages.

Avantages

La recherche des deuxièmes utilisateurs s'effectue de manière focalisée ; elle ne s'effectue pas en fonction de leur profil global d'intérêts (vis-à-vis du profil global

d'intérêts du premier utilisateur). Ceci constitue un avantage tant au niveau de la pertinence du résultat que des performances d'exécution.

Au niveau de la pertinence du résultat, l'avantage de la focalisation est de pouvoir comparer les profils des utilisateurs en faisant abstraction des éléments de ces profils qui ne sont pas dans le contexte qui intéresse l'utilisateur au moment courant. Autrement-dit, le nombre des éléments pertinents (par rapport au contexte courant) qui se trouvent être en commun entre le premier utilisateur et un deuxième utilisateur avec lequel il est comparé, d'une part n'est pas « dilué » dans la masse des éléments qui sont en commun globalement, et d'autre part peut être relativisé par rapport à la taille du domaine du contexte courant.

Pour focaliser les recherches de profils voisins, l'état de la technique connue consiste à partitionner les éléments des profils selon des catégories adoptées par l'ensemble des utilisateurs, qui permettent de les classer globalement ou par rapport à des attributs qui les caractérisent. On spécifie ainsi leur appartenance aux domaines respectifs pris en compte lors des recherches, ce qui permet d'élaguer les éléments des profils qui ne sont pas dans le domaine d'intérêt courant de l'utilisateur. Cependant, la catégorisation des éléments des profils nécessite pour les utilisateurs de s'entendre sur les vocabulaires de catégories à partager. Or cette approche n'est pas avantageusement adaptée à des systèmes largement décentralisés tels que l'Internet. L'avantage essentiel du procédé selon cet aspect de la présente invention est de focaliser la recherche sans devoir classer les éléments des profils selon des catégories partagées, et donc sans avoir à s'entendre sur des vocabulaires communs de catégories.

Au niveau des performances, la sélection peut se faire sur la base d'une recherche en accès direct grâce à une structure de données (telle qu'un fichier inversé) qui permette de retrouver les deuxièmes utilisateurs proches directement à partir de ladite page, ou même à partir de ladite page et de ses liens ajoutés.

Détails de la détection de Répertoires Proches

Comme les utilisateurs peuvent placer des mêmes liens ajoutés dans des répertoires différents, le procédé peut servir à sélectionner des répertoires au lieu d'utilisateurs. Notamment, le système peut avantageusement servir à sélectionner ;

- des répertoires « proches » du premier utilisateur,

- plusieurs répertoires d'un même deuxième utilisateur.

Dans la suite, on décrira donc l'approche consistant à sélectionner des répertoires plutôt que des utilisateurs.

Comme déjà indiqué pour le cas de la recherche des utilisateurs les plus proches, la sélection des répertoires les plus proches peut également se faire très efficacement grâce à une structure de données (tel qu'un fichier inversé) permettant de retrouver en accès direct les répertoires contenant une page donnée (la page courante de l'utilisateur). L'accès direct peut même se baser sur la page donnée et les liens Ajoutés qui lui sont associés.

On notera à cet égard que les liens Ajoutés à partir desquels est lancée la sélection sont ceux du répertoire courant (ou ceux de l'ensemble des répertoires courants sélectionnés par l'utilisateur, comme décrit dans la section « Création de liens Ajoutés dans des Répertoires »).

La figure 57 schématise les deux types d'exécution de requête d'accès aux données (dans le cas où les données sont rangées dans des tables relationnelles) :

- l'accès direct aux liens Ajoutés à partir :
  - d'une ou plusieurs pages,
  - d'un utilisateur,

- dans un (ou plusieurs) répertoires ;

- l'accès direct aux utilisateurs et leurs répertoires à partir :
  - d'une (ou plusieurs) pages,
  - et de ses liens ajoutés (obtenus par une requête du premier type).

Rappelons que le premier type de requête est celui permettant les fonctionnalités « Retrouver des liens Ajoutés » décrites plus haut, alors que le deuxième type de requête est celle de « Détection de Répertoires Proches », objet de la présente section.

Les répertoires les plus proches, sélectionnés automatiquement, sont présentés à l'utilisateur dans la deuxième liste (intitulée « voisins d'intérêt ») illustrée de manière schématique dans la figure 55a. Comme déjà décrit dans la section précédente (« Publication de liens ajoutés »), l'utilisateur peut cocher un ou plusieurs des répertoires qui lui sont proposés par le système. En résultat, de nouveaux liens ajoutés, puisés dans ce ou ces répertoires, lui sont suggérés en association avec la page courante (conformément à la description de la section précédente).

Le système peut aussi offrir à l'utilisateur des moyens de tenter de se mettre en relation – par des moyens de communication synchrone sur Internet tels que la messagerie simultanée, la visioconférence, etc. – avec les utilisateurs possédant ces répertoires qui seraient en ligne en même temps que l'utilisateur.

En effet, dans les deux types d'architecture client-serveur représentés respectivement sur les figures 44 et 45, les requêtes que les utilisateurs forment au moyen de leur navigateur, sont communiquées au serveur dans lequel est exécuté le système. Ce dernier est donc capable de savoir si les utilisateurs en question sont en ligne et si on peut communiquer avec eux.

L'utilisateur peut aussi se mettre en relation par communication asynchrone, notamment par courrier électronique ou dans le cadre de forums de discussion, avec ces utilisateurs « proches » (que lui fait « découvrir » le système), même s'ils ne sont pas en ligne en même temps.

On notera à cet égard, que grâce aux moyens déjà mentionnés de sélection en accès direct d'utilisateurs à partir de pages qui les intéressent (en plus d'une structure de chemin d'accès direct aux pages à partir d'utilisateurs et répertoires) le système est apte à construire, avec de très bonnes performances, des communautés d'utilisateurs partageant des mêmes centres d'intérêts. Les sujets de discussion créés sont ceux pour lesquels un ensemble de pages de taille suffisante a intéressé un grand nombre d'utilisateurs. Ces derniers sont alors invités à participer à un forum de discussion ayant comme thème cet ensemble de pages.

On va maintenant décrire la méthode de calcul de proximité.

En partant de la page courante (c'est-à-dire la page visualisée par le premier utilisateur) avec ses liens ajoutés dans le répertoire courant – ou encore avec l'union des liens ajoutés associés à cette page dans les répertoires courants sélectionnés par l'utilisateur – le système trouve parmi les autres répertoires (du premier utilisateur et/ou parmi les répertoires publiés de seconds utilisateurs) contenant ladite page, ceux qui ont le score de proximité le plus élevé autour de ladite page.

En résultat, le système propose à l'utilisateur ces répertoires dans l'ordre de leur proximité.

La proximité est calculée à différents niveaux.

Au premier niveau, la proximité est fonction du nombre de liens ajoutés communs associés à la page courante. La figure 58 illustre la détermination de la proximité au niveau 1.

Au niveau 2, la proximité est également fonction du nombre de liens ajoutés communs associés aux pages reliées à la page courante par lien ajouté, comme l'illustre la figure 59.

Au niveau n, la proximité est fonction du nombre de liens ajoutés communs associés aux pages reliées indirectement à la page courante par lien ajouté, n étant le nombre d'étapes d'indirection.

Proximité transitive

On va décrire ci-après un perfectionnement du procédé de calcul de proximité entre deux répertoires décrit plus haut.

Le système peut enrichir son évaluation de proximité en anticipant les propagations probables de liens ajoutés dans le futur.

En effet, les liens ajoutés qui au moment courant ne sont pas encore propagés (suggérés) d'un répertoire proche à un autre, le seront probablement dans le futur, et, après acceptation éventuelle, seront prêts à être suggérés à un troisième répertoire (et ainsi de suite). Le système peut enrichir son estimation de proximité en anticipant ces propagations (suggestions) et acceptations, et en calculant la proximité d'un répertoire en prenant en compte cette anticipation.

Ceci peut se faire par deux approches différentes :

- soit par simulation prédictive :

La simulation prédictive prend en compte la probabilité d'acceptation (telle que décrite plus loin) des futures suggestions de liens ajoutés, dans les répertoires proches du répertoire courant de l'utilisateur.

Le procédé de simulation prédictive consiste à simuler la propagation de liens ajoutés, à partir de répertoires à priori non proches du répertoire courant, vers des répertoires proches du répertoire courant. Suite à la propagation (simulée) d'un lien ajouté à partir de chaque répertoire non proche (vers un répertoire proche), un coefficient de probabilité d'acceptation (décrit plus loin) lié à ce répertoire non proche est associé audit lien ajouté.

La proximité peut alors être affinée en tenant compte des nouveaux liens ajoutés (introduits par la simulation) pondérés par leur coefficient de probabilité d'acceptation.

Le calcul de proximité peut être affiné progressivement, en prenant en compte des niveaux de plus en plus indirects. En effet, la simulation prédictive peut se faire à des profondeurs d'indirection de plus en plus grandes : ainsi un lien ajouté peut être propagé (par simulation) dans un répertoire proche du répertoire proche en question, avant d'arriver dans ce dernier, et ainsi de suite. Pour chaque nouveau répertoire, le système applique au lien ajouté le coefficient de probabilité d'acceptation associé au répertoire d'où il a été puisé. A l'arrivée dans le répertoire proche en question, les liens ajoutés (suggérés par simulation) ont un coefficient de probabilité d'acceptation qui représente la composition des probabilités d'acceptation dans les différents répertoires où ils sont passés.

Optionnellement, l'utilisateur peut paramétrer le système pour lui demander de lui suggérer directement les liens ajoutés propagés par simulation dans le répertoire proche (bien que ces liens ajoutés ne soient pas encore acceptés par le possesseur du répertoire proche).

De manière progressivement plus indirecte, l'utilisateur peut demander au système de lui suggérer directement les liens ajoutés propagés par simulation dans les répertoires proches d'un répertoire proche, et ainsi de suite. Lors des propagations

d'un répertoire à l'autre, les coefficients de probabilité d'acceptation sont composés et les suggestions peuvent être effectuées dans l'ordre de ces coefficients.

- soit selon l'approche décrite ci-dessous et appelée calcul de « proximité transitive » et qui offre l'avantage d'une plus grande rapidité de calcul.

Considérons trois répertoires R1,R2 et R3. La « proximité transitive de R3 pour R1 » est fonction du minimum entre « la proximité de R2 pour R1 » et « la proximité de R3 pour R2 ».

Le principe sous-jacent est, en effet, que dans une chaîne de propagations potentielles de liens ajoutés, la proximité transitive est fonction de la proximité du maillon le plus faible de la chaîne.

A la valeur obtenue (en prenant le minimum des proximités), on doit aussi appliquer la composition des coefficients de probabilité d'acceptation des répertoires respectifs (acceptation de R2 par R1 et de R3 par R2).

Un exemple est illustré sur la figure 60 :

- la proximité de R2 pour R1 est de 2 ; en effet les liens ajoutés pointant sur les pages p2 et p3 sont en commun ;

- la proximité de R3 pour R2 est de 1 (p5 seul est en commun) ; il en découle que la Proximité Transitive de R3 pour R1 est de 1 (le minimum entre 2 et 1), alors que la proximité de R3 pour R1 n'est que de 0 (aucun lien ajouté en commun).

Bien évidemment, la proximité transitive peut aussi être calculée dans une chaîne plus longue de propagations potentielles, en prenant le minimum des proximités entre les répertoires pris séquentiellement dans ladite chaîne et en lui appliquant la

composition des coefficients de probabilité d'acceptation existant pour tous les maillons de la chaîne.

Coefficient de probabilité d'acceptation

La détermination de la probabilité d'acceptation, dans un premier répertoire, d'un lien ajouté suggéré provenant d'un deuxième répertoire peut se baser sur l'historique des acceptations de liens ajoutés suggérés à partir de ce deuxième répertoire.

Dans ce sens, chaque premier répertoire maintient un compteur pour chaque deuxième répertoire à partir duquel il a reçu des suggestions dans le passé. Bien entendu, ce compteur peut être maintenu pendant une durée limitée dans le temps. La valeur de ce compteur représente le coefficient de probabilité d'acceptation.

Le coefficient de probabilité d'acceptation est fonction du nombre de liens ajoutés acceptés (en mode normal ou en mode gelé) moins le nombre de liens ajoutés refusés rapporté au nombre (total) de liens ajoutés suggérés.

Ce procédé peut comporter diverses sophistications que saura mettre en œuvre l'homme du métier, et notamment une pondération permettant de donner un poids plus fort aux acceptations/refus plus récents.

Pour un nouveau répertoire, c'est-à-dire à partir duquel aucun lien ajouté n'avait été suggéré dans le passé vers le répertoire courant de l'utilisateur, le système peut se baser sur un répertoire intermédiaire (ayant un historique dans le répertoire courant et le nouveau répertoire ayant un historique chez lui) pour déterminer la probabilité d'acceptation. Ceci s'effectue selon l'approche suivante :

Si un répertoire R1 jouit d'un fort coefficient de probabilité d'acceptation (C0-1) auprès d'un répertoire R0, et qu'un répertoire R2 a un fort coefficient de probabilité

d'acceptation (C1-2) auprès du répertoire R1, alors, par transitivité, R2 aura un fort coefficient de probabilité d'acceptation (C0-2) auprès du répertoire R0.

Plus précisément, la valeur C0-1 reflète le degré de confiance que le possesseur du répertoire R0 accorde au contenu du répertoire R1. En effet, puisqu'en général le possesseur du répertoire R0 accepte les suggestions provenant du répertoire R1, il accorde un degré de confiance relativement fort à la validité de l'acte d'acceptation dans le répertoire R1. En d'autres termes, le possesseur du répertoire R0 considère que le possesseur du répertoire R1 a un bon jugement en matière d'acceptation et de refus dans ledit répertoire R1.

Or il se trouve que le possesseur du répertoire R1 a accepté dans ce répertoire une grande partie des suggestions provenant du répertoire R2 (c'est-à-dire que la valeur C1-2 est élevée). Puisque son jugement est bon aux yeux du possesseur du répertoire R0, le système peut conclure que le possesseur du répertoire R0 va probablement aussi accepter les suggestions provenant du répertoire R2.

Par le même raisonnement, si la valeur C0-1 est forte et si la valeur C1-2 est faible, alors, le système peut conclure que la valeur C0-2 sera faible.

Bien évidemment, la profondeur de la transitivité (le nombre des intermédiaires) peut être plus grande : le système peut se baser sur plusieurs intermédiaires qui se sont fortement validés dans le passé (de préférence le passé récent - voir plus haut à propos de la pondération) pour prédire qu'un nouveau répertoire sera probablement validé ou pas. Mais, bien évidemment, la fiabilité du coefficient obtenu baissera avec la profondeur de la transitivité.

On a déjà mentionné que la détection de répertoires proches permet de présenter ces répertoires à l'utilisateur dans la deuxième liste (intitulée « voisins d'intérêt ») illustrée de manière schématique dans la figure 55a. Le coefficient de probabilité d'acceptation est un indicateur qui est présenté avec chacun de ces répertoires (il peut

par exemple être intitulé « statistiques d'acceptation »). L'utilisateur préférera cocher des répertoires qui ont ce coefficient le plus élevé possible.

Section 7 - Suggestion de liens ajoutés par l'administrateur d'un site

Présentation

L'administrateur d'un site (« Webmaster » selon la terminologie anglo-saxonne) spécifie, dans le code source de la page à laquelle accède un utilisateur, ou encore dans une table contenue dans un serveur approprié et dont la clé est l'adresse de ladite page, l'ensemble des adresses des répertoires (dits « Spécialistes ») qui peuvent être proposés à l'utilisateur dans la première liste de la figure 55a.

Les deux étapes suivantes peuvent être mises en œuvre dans l'exploitation de ces répertoires (répertoires candidats) pour un utilisateur donné :

1. En première étape, le système ne retient parmi les répertoires candidats que ceux qui sont les plus proches du répertoire courant (au sens de la section précédente « Détection de Répertoires Proches »). Toutefois, cette approche n'est valide que si la page à laquelle l'utilisateur a accédé n'est pas nouvelle pour celui-ci, et que l'utilisateur lui avait déjà associé des liens ajoutés, de manière à ce que le système puisse les comparer avec ceux figurant dans chacun des répertoires candidats spécifiés par l'administrateur (pour déterminer dans quelle mesure ce répertoire est proche, et ne retenir ainsi que les répertoires candidats les plus proches).

Si, au contraire, la page à laquelle l'utilisateur accède est nouvelle pour lui, le système peut, optionnellement, tenter tout de même de sélectionner des répertoires candidats proches, en comparant les liens ajoutés qui se trouvent dans le répertoire courant de l'utilisateur (ces liens ajoutés n'étant pas associés avec la page courante, puisque celle-ci est nouvelle, mais à d'autres pages) avec les liens ajoutés des

répertoires candidats, en partant de pages situées à des niveaux progressivement plus éloignés de la page courante dans les répertoires candidats.

En cas d'échec, le système peut proposer les premiers répertoires spécifiés par l'administrateur du site comme devant être présentés par défaut.

2. Le système ne retient, dans les répertoires choisis en première étape, que les liens ajoutés les plus avantageux selon un ensemble de critères de pertinence dont le poids relatif peut être réglé par l'administrateur du site.

A partir des répertoires Spécialistes candidats (et/ou des répertoires proches « voisins d'intérêt »), le système peut aussi synthétiser un premier répertoire spécialiste de manière « personnalisée » et le proposer à l'utilisateur par défaut. Le contenu de ce nouveau répertoire est créé en regroupant des liens ajoutés sélectionnés dans les répertoires candidats, qui à priori sont potentiellement les plus pertinents pour l'utilisateur.

En relation avec chaque lien ajouté, l'administrateur peut spécifier :

- la manière dont ledit lien ajouté va être graphiquement présenté avec la page courante ;

- et la manière dont le lien ajouté inverse sera graphiquement présenté quand la page vers laquelle pointe ledit lien ajouté pointe va être visualisée.

Répertoires de la liste intitulée « Spécialistes »

Le document source d'une page à laquelle accède l'utilisateur (par exemple une page écrite en langage HTML) peut contenir, parmi les données non affichées à l'écran, l'adresse d'un premier répertoire (contenant des liens ajoutés) à proposer par défaut à l'utilisateur.

Cette adresse (composée par exemple du pseudonyme de l'utilisateur et du chemin du répertoire que l'administrateur du site a choisis) est utilisée par le système pour être présentée en première position dans la première liste (« Spécialistes ») de la figure 55a.

Cette adresse est cochée (sélectionnée) par défaut par le système, ce qui signifie que l'utilisateur est implicitement considéré comme souhaitant se faire suggérer les liens ajoutés du répertoire en question dans sa propre liste de liens ajoutés (conformément à la description faite dans la section « Publication de liens ajoutés »).

Dans le document source, l'administrateur peut aussi spécifier des adresses supplémentaires de répertoires d'autres spécialistes. Celles-ci serviront d'éléments optionnels dans la première liste (« Spécialistes) » de répertoires proposés à l'utilisateur, mais ne seront pas cochées par défaut.

Alternativement, ces adresses de répertoires spécialistes, au lieu d'être notées dans le document source de la page à laquelle accède l'utilisateur, peuvent être stockées sur un serveur dans une table qui fournit la correspondance entre l'adresse de la page et les adresses des répertoires Spécialistes qui lui sont attribués. Dans le cas où ledit serveur est aussi le serveur de liens ajoutés pour l'utilisateur courant, cette alternative permet d'éviter un aller-retour supplémentaire sur le serveur de liens ajoutés dans le cas de l'architecture de la figure 44.

Sélection parmi les répertoires candidats

Le nombre d'adresses de répertoires Spécialistes (spécifiées par l'administrateur) peut être bien plus élevé que le nombre de répertoires qui seront effectivement présentés à l'utilisateur. Un nombre maximal de répertoires à présenter peut être fixé par l'administrateur et/ou par l'utilisateur.

Comme déjà mentionné brièvement dans la présentation faite plus haut, parmi les répertoires candidats, le système choisit les répertoires les plus proches de l'utilisateur. Ceci s'effectue :

- soit directement selon la technique décrite plus haut dans la section « Détection de Répertoires Proches »,

- soit selon une extension de cette technique, appelé « Proximité Décalée », que l'on va maintenant décrire.

Proximité Décalée

Dans le cas où la page courante (visualisée par l'utilisateur) :

- ne possède aucun lien ajouté (notamment parce que l'utilisateur y accède pour la première fois dans son répertoire courant)

- ou n'a pas suffisamment de liens ajoutés en commun avec aucun des répertoires candidats(par rapport à un critère quantitatif fixé lors du paramétrage du système),

le système, n'ayant pas la possibilité de détecter (par le procédé décrit dans la section « Détection de Répertoires Proches »), parmi les répertoires candidats, les répertoires « les plus proches », sélectionne les répertoires candidats par les étapes suivantes :

a. le système retient les répertoires candidats offrant le plus grand nombre de pages d'éloignement de niveau 1 (c'est-à-dire des pages pointées par des liens ajoutés associés à la page courante) qui soient en commun,

b. pour chacune des pages d'éloignement de niveau 1 qui sont en commun, le système calcule la Proximité (tel que décrit dans la section « Détection de Répertoires Proches ») en partant de ces pages.

La Proximité Décalée pour le répertoire candidat est fonction des Proximités ainsi déterminées.

Au final, le système choisit les répertoires candidats offrant les valeurs les plus élevées de proximité Décalée déterminées comme décrit ci-dessus.

Dans le cas où le résultat de la recherche du point « a. » n'est pas suffisamment satisfaisant, le système peut éventuellement rechercher, dans le répertoire courant, des pages situées à des niveaux progressifs d'éloignement dans les répertoires candidats (Proximité Décalée d'éloignement > 1).

Au point « b. », le niveau de profondeur de recherche de Proximité peut éventuellement être augmenté ou diminué (le cas de recherche le moins coûteux en ressources informatiques étant celui avec une profondeur de niveau 0, c'est-à-dire en court-circuitant le point « b. »).

La figure 61 illustre la détermination de proximité décalée d'éloignement 1 pour un répertoire candidat. Dans cet exemple, le système trouve, dans le répertoire courant de l'utilisateur :

- deux pages en commun (p1 et p2),
- pour la page p1 du répertoire candidat, une valeur de proximité égale à deux liens ajoutés sur trois ; et pour la page p2, une valeur de proximité égale à un lien ajouté sur deux.

On notera que cette technique de calcul de proximité décalée à partir d'un niveau progressif d'éloignement en partant de la page courante peut également être utilisée de manière générale pour rechercher des répertoires proches (notamment parmi un ensemble d'utilisateurs), en complément de la technique de calcul de proximité décrite dans la section « Détection de Répertoires Proches ».

La figure 62 illustre ainsi la détermination de proximité décalée d'éloignement 1 d'un répertoire B par rapport à une page courante p0 dans un Répertoire A. On observe que dans cet exemple, la détermination de proximité tel que décrite dans la section « Détection de Répertoires Proches ») aurait échoué puisque la page p0 n'existe pas dans le répertoire B.

Sélection parmi les liens ajoutés

Les liens ajoutés contenus dans les répertoires candidats sélectionnés (c'est-à-dire les « Spécialistes » proposés dans la première liste de la figure 55a) peuvent comporter bien plus de liens ajoutés (en mode « Accepté » ou « Gelé ») que ceux qui seront suggérés aux utilisateurs. En effet, le système est apte à sélectionner les liens ajoutés les plus pertinents pour chaque utilisateur qui les recevra. Ainsi la suggestion de liens ajoutés se fait de manière « personnalisée ». La sélection est effectuée par exemple en fonction d'un ou plusieurs des critères suivants (sélectionnés notamment en fonction des ressources qu'ils nécessitent à priori pour leur mise en œuvre):

- l'intérêt potentiel du lien ajouté en question (dit « lien ajouté candidat ») pour l'utilisateur : comme déjà mentionné dans la section « Création de liens ajoutés dans des Répertoires », les liens ajoutés sont bidirectionnels, c'est-à-dire qu'un lien ajouté associé à une page P1 et pointant sur une page P2 implique (dès qu'il est Accepté) la création du lien ajouté associé à la page P2 et pointant sur la page P1 ; il est donc dans l'intérêt de l'administrateur du site de suggérer un lien ajouté sur une page P2 potentiellement la plus intéressante possible, puisque

    * plus la page P2 est intéressante, plus l'utilisateur va la consulter ;

    * et à chaque fois qu'il va la consulter, le lien ajouté pointant sur la page P1 va y figurer et va ainsi inciter l'utilisateur à aller consulter ladite page P1.

Pour satisfaire à ce critère particulier, le système a notamment la possibilité d'analyser si le lien ajouté candidat avait été accepté ou gelé dans des répertoires proches du répertoire courant de l'utilisateur. Pour ce faire, et comme l'illustre la

figure 63, le système recherche des répertoires (répertoire proche candidat) possédant :

* la page courante (à noter que, du fait que la page courante est nouvelle pour l'utilisateur, aucun lien ajouté n'y est encore associé) ;

* le lien ajouté candidat associé à la page courante, en mode accepté ou gelé,

* et le répertoire de l'utilisateur ayant une forte valeur de proximité décalée d'éloignement 1 (ou progressivement plus si nécessaire) par rapport au répertoire proche candidat en partant de la page courante.

- le fait que l'utilisateur avait déjà reçu la page vers laquelle pointe le lien ajouté candidat (s'il ne s'agit pas d'une nouvelle page pour lui) dans le répertoire courant, et le fait qu'en outre la page pointée par le lien ajouté candidat est déjà bien « ancrée » dans le répertoire courant (dans le sens où l'utilisateur a une probabilité relativement forte de la consulter en navigant d'une page a l'autre du répertoire), étant donné que plus l'utilisateur accédera à la page pointée par le lien ajouté, plus il verra le lien ajouté inverse qui l'incitera à aller revisiter la page courante (voir cependant le troisième sous-critère ci-dessous qui consiste à fixer un seuil supérieur au nombre de liens ajoutés préexistants) ; la propriété d'ancrage est mesurée en comptant dans le répertoire courant le nombre de liens ajoutés qui pointent sur la page en question (ou plus simplement, puisque les liens ajoutés sont bidirectionnels, en comptant les liens ajoutés associés à la page en question).

- dans quelle mesure l'utilisateur avait déjà accepté ou gelé cette page (ou au contraire, combien de fois il l'avait laissée en mode suggéré ou il l'avait refusée) : ainsi, si l'utilisateur avait déjà reçu cette page (pointée par le lien ajouté candidat) dans le répertoire courant, et si de plus il l'avait acceptée (ou gelée), ceci indique que cette page l'intéresse et qu'elle sera donc probablement à nouveau acceptée en tant que lien ajouté de la page courante ; il est donc plus avantageux de choisir le lien ajouté pointant sur cette page plutôt qu'un lien ajouté pointant sur une page nouvelle pour l'utilisateur ; en revanche, si l'utilisateur avait déjà reçu cette page mais depuis ce moment il l'a laissée en mode suggéré, ou, à fortiori, il l'a refusée, le système

préférera ne pas la choisir, car le lien ajouté candidat a relativement moins de chances d'être accepté par l'utilisateur.

- le nombre de liens ajoutés couramment associés à la page pointée par le lien ajouté candidat : si, dans le répertoire courant de l'utilisateur, une multitude de liens ajoutés sont déjà associés à la page pointée par le lien ajouté candidat, le lien ajouté inverse au lien ajouté candidat sera « noyé » dans cette multitude et n'aura pas l'effet d'incitation souhaité ; il n'est donc pas toujours avantageux de choisir un lien ajouté candidat dont la page vers laquelle il pointe possède déjà de trop nombreux liens ajoutés dans le répertoire de l'utilisateur (on notera que ce sous-critère est l'inverse du critère d'ancrage ; il doit être utilisé pour fixer un seuil qu'il ne faut pas dépasser) ;

l'administrateur indique alors avantageusement, avec chaque répertoire spécialiste (qu'il spécifie par exemple dans la partie non affichée du document source de la page, ou encore dans une table sur le serveur, comme décrit plus haut) :

* le nombre maximum de liens ajoutés à suggérer (voir plus haut), étant rappelé que l'utilisateur aussi peut contraindre ce nombre ;

* le poids relatif à donner à chacun des critères (c'est-à-dire la formule d'agrégation pondérée des critères) décrits ci-dessus, ainsi qu'éventuellement des indications sur les heuristiques à mettre en œuvre dans leur prise en compte (il s'agit notamment d'éviter les critères qui consommeraient trop de ressources).

Suggestion directe dans le Répertoire par défaut

Le système peut synthétiser et proposer des liens ajoutés à partir :

- des répertoires Spécialistes et/ou
- des répertoires proches déterminés automatiquement par le système

et présenter les liens ajoutés obtenus à partir de ces répertoires (répertoires candidats) par exemple à travers le premier répertoire de la première liste de la figure 55a, qui est proposé à l'utilisateur par défaut.

Ce premier répertoire est ainsi « synthétisé » de manière différente pour chaque utilisateur.

Pour ce faire, dans les répertoires candidats, le système choisit les liens ajoutés :

- qui pointent sur une page,
  * alors que cette page est déjà existante et a été acceptée (ou gelée) un nombre de fois suffisant dans le répertoire courant de l'utilisateur
  * ou qui s'est avérée être intéressante chez un nombre suffisant d'utilisateurs voisins d'intérêt (ou plus exactement, qui fut acceptée ou gelée dans un nombre suffisant de répertoires proches),
- et optionnellement, qui pointent sur une page pour laquelle la « Proximité Décalée Cumulée » d'éloignement 1 (à partir de la page courante), obtenue en additionnant les Proximités Décalées d'éloignement 1 dans les répertoires candidats, soit la plus grande possible.

Présentation graphique

Avec chaque lien ajouté, l'administrateur du site peut spécifier sous quelle forme, avec quelle image et à quelle position de la page courante, le lien ajouté devra être posé, la présentation pouvant être par exemple :

- sur le bord de la page (dans une liste de « vignettes »),
- sur la page (comme une affichette adhésive), ou sous la forme d'une petite icône située sur la page qui se transforme en une telle affichette au passage de la souris,
- ou encore dans des régions de la page qui étaient prévues pour contenir de la publicité.

L'administrateur peut aussi spécifier ces paramètres de présentation graphique pour le lien ajouté inverse (qui sera affiché sur la page vers laquelle pointe le lien ajouté en question).

En particulier, l'administrateur peut souhaiter expressément spécifier à quelle position ou dans quelle région, sur la page vers laquelle pointe le lien ajouté en question, sera appliquée la représentation graphique du lien ajouté inverse. C'est en effet par ce moyen qu'il pourra mieux inciter l'utilisateur à venir consulter sa propre page (la page courante). Son audience dépendra donc en partie de la position (ainsi que l'apparence graphique) du lien ajouté inverse.

La figure 64 illustre la présentation graphique d'un lien ajouté inverse sur une page (sous forme d'affichette). Une page p1 a un lien ajouté sur une page p2 qui est déjà très encombrée d'une multitude de liens ajoutés présentés graphiquement sous forme de vignettes (dans une liste verticale). Sur la page p2, le lien ajouté inverse pointant sur p1 n'est pas présentée comme une vignette de plus, mais sous forme d'une affichette directement sur le contenu de la page. La représentation graphique du lien ajouté inverse peut aussi être automatiquement inséré dans une région qui normalement aurait dû contenir de la publicité. L'administrateur espère par là attirer l'attention de l'internaute.

Section 8 - Calcul automatique de prévision d'Audience

Le service offert par le système, consistant à permettre à l'administrateur d'inciter l'utilisateur à cliquer sur un lien ajouté inverse (pointant sur la page publiée par ledit administrateur) :

- peut être proposé (fonction de « devis ») au moyen d'un calcul de prévision d'audience ;

- peut être facturé (fonction « facturation ») pour le service effectivement rendu, sur la base :

* de l'audience de la page à laquelle est associé le lien ajouté inverse (le nombre de fois que la page p2 de la figure 64 est vue par les utilisateurs du système),

* des clics effectifs, c'est-à-dire en fonction du nombre de clics effectivement réalisés sur les liens ajoutés inverses.

En effet, non seulement, un lien ajouté inverse incite directement l'utilisateur à accéder à la page vers laquelle il pointe (en l'occurrence la page p1 de la figure 64), mais en outre, les liens ajoutés de la page (p2) à laquelle le lien ajouté inverse est associé peuvent être publiés (voir la section « Publication de liens ajoutés ») et être ainsi propagés d'un utilisateur à l'autre.

Dans ce schéma de propagation, seul les liens ajoutés inverses acceptés par un utilisateur seront suggérés aux utilisateurs qui sont en l'aval dans la chaîne de propagation.

La prévision d'audience peut être effectuée avantageusement en appliquant le coefficient de probabilité d'acceptation qui tient compte du comportement des utilisateurs (tel que décrit plus haut dans la section « Détection de Répertoires Proches »).

Section 9 - Publication de Répertoires

Un perfectionnement du système consiste à permettre à l'utilisateur de publier (ou mettre à la disposition d'un groupe restreint) au moins un sous-ensemble des répertoires de sa toile personnelle.

De la même manière que dans l'approche décrite plus haut dans la section « Publication de liens Ajoutés », les liens vers des pages contenus dans un répertoire publié pourront être visualisés :

- directement au moyen d'un navigateur standard (par exemple suite à la réception d'un e-mail) ;
- ou au moyen du système.

Le répertoire consulté par l'utilisateur au moyen du système devient son répertoire courant. Il peut s'agir d'un de ses propres répertoires ou d'un répertoire publié par un autre utilisateur. L'utilisateur peut « manuellement » insérer de nouveaux liens dans ce répertoire.

L'utilisateur qui visualise le contenu d'un répertoire au moyen du système, peut également enrichir ce contenu en se faisant suggérer des liens à partir d'autres répertoires qui lui sont proposés dans les listes « Spécialistes », « Voisins d'intérêt », et « Copains » (comme pour le cas des liens Ajoutés). Ces trois listes sont rappelées ci-dessous :

1. Les répertoires Spécialistes sont ceux proposés par l'administrateur du site, et celui-ci peut proposer à l'utilisateur un nombre de répertoires Spécialistes (candidats) plus grand que le nombre de répertoires qui seront présentés dans cette première liste. Le système sélectionne :

- d'abord les répertoires spécifiés comme devant être présentés par défaut (s'il en existe),

- ensuite, au fur et à mesure de l'acceptation de liens par l'utilisateur (l'acceptation est décrite plus loin dans cette section) dans le répertoire courant, le système sélectionne les répertoires dont les liens sont les plus voisins possibles des liens couramment déjà acceptés (ou gelés) par l'utilisateur.

Pour ce faire, le système choisit essentiellement les répertoires contenant le plus de liens en commun avec l'ensemble de liens acceptés (ou gelés) par l'utilisateur au moment courant.

Un répertoire Spécialiste peut contenir un nombre de liens supérieur au nombre de liens qui seront suggérés à l'utilisateur. Le système sélectionne alors de préférence les liens associés par le plus grand nombre de liens ajoutés (dans le répertoire Spécialiste candidat) avec des liens qui (dans le répertoire courant) sont déjà acceptés. Bien évidemment, ne seront suggérés à l'utilisateur que les liens qui ne figurent pas déjà dans le répertoire courant.

2. Les répertoires de Voisins d'intérêt sont déterminés automatiquement par le système. Ce sont :

- soit des répertoires proches et de même catégorie, qui sont déterminés en amenant le système à sélectionner, parmi les répertoires de même catégorie, ceux qui contiennent le plus de liens en commun avec les liens acceptés (ou gelés) par l'utilisateur.

- soit des répertoires contenant le plus de liens en commun avec des liens acceptés (ou gelés) dans le répertoire courant, quelles que soient leurs catégories ; cette approche peut se révéler performante si la sélection de ces répertoires peut se faire en accès direct à partir des liens contenus dans le répertoire courant (voir le type de requête schématisé sur la figure 65) ; en outre, cette approche permet d'associer des catégories aux répertoires automatiquement, car les catégories associées à chaque répertoire déterminé automatiquement sont proposées à l'utilisateur, lorsque ce dernier sélectionne ledit répertoire.

Le grand avantage de ce procédé est qu'il permet aux utilisateurs d'associer aux répertoires des catégories appartenant à un vocabulaire commun (le système incite à parler un « langage commun »).

3. Les répertoires Copains sont les répertoires choisis par l'utilisateur (Voir la section « Publication de liens Ajoutés »).

Le principe de Publication et Suggestion étant le même que pour la Publication de liens Ajoutés (voir la section « Publication de liens Ajoutés »), nous ne précisons pas ici tous les détails, mais quelques particularités des liens (associés aux répertoires) par opposition aux liens ajoutés (associés aux pages).

Dans l'ensemble de liens (vers des pages) qui sont contenus dans le répertoire que l'utilisateur est en train de visualiser, l'utilisateur reçoit des liens supplémentaires à partir des répertoires sélectionnés dans les trois listes sus-décrites (Spécialistes, Voisins, Copains).

Comme pour les liens Ajoutés, le système peut avantageusement comporter des moyens pour sélectionner des liens sur la base de critères (« Offreur », « Demandeur », etc), données par l'utilisateur ou associés au répertoire visualisé.

Les liens qui sont ainsi suggérés à l'utilisateur doivent être distingués de ceux qu'il avait lui-même ajoutés : il peut en effet en refuser certains. Dans ce but, les liens peuvent se trouver dans un parmi plusieurs modes différents: « Suggéré », « Accepté », « Refusé » (et « Gelé », ce dernier mode étant décrit dans la section « Liens en mode Gelé »).

Les liens reçus par un premier utilisateur, à partir des liens publiés par un deuxième utilisateur (ou à partir d'un des répertoires du premier utilisateur), sont chez le premier utilisateur au départ en mode « Suggéré ».

Suite à une action de l'utilisateur, chaque lien ajouté en mode Suggéré peut passer à l'un parmi les autres modes. L'interface utilisateur permettant la mise en œuvre de ces changements de mode est schématisée sur la figure 66.

Cette figure illustre également le fait que, les répertoires étant visualisés comme des pages, l'utilisateur peut leurs associer des liens ajoutés. On voit ainsi qu'au répertoire r1 ont été associés (par la technique des liens ajoutés) les répertoires r2 et r3 ainsi que la page p6.

Les moyens de stockage associent donc aux liens un attribut « Mode » qui peut prendre comme valeur : Accepté, Refusé ou Gelé. La figure 56 présentait un diagramme de classe pour les liens Ajoutés, selon la représentation standard UML (« Unified Modeling Language »), mettant en évidence l'attribut Mode des liens Ajoutés. Nous complétons cette description avec la figure 67 qui montre que les liens (vers les pages) possèdent également un attribut Mode. Cet attribut peut lui aussi prendre la valeur Accepté, Refusé ou Gelé. De plus, la classe « Répertoire » comporte l'attribut « Catégories » qui a pour fonction de permettre de comparer des répertoires de même catégorie (indépendamment des noms que les utilisateurs leur ont donné).

Un lien en mode Suggéré n'existe que pendant la visite d'un répertoire sélectionné dans l'une des trois listes déjà mentionnées. Il n'est pas affiché en dehors de cette visite.

Un lien en mode Suggéré n'est pas stocké dans l'espace de stockage personnel de l'utilisateur car, à chaque nouvelle visite, il est recréé à partir du lien correspondant stocké dans le répertoire source.

Le passage d'un lien du mode Suggéré au mode Refusé a pour effet de filtrer automatiquement ledit lien lors des visites ultérieures.

Le passage au mode Accepté signifie que l'utilisateur « valide » le lien qui est en mode Suggéré. Cette action a pour effet :

1. de stocker ledit lien dans son espace de stockage personnel, de manière à ce qu'il lui soit présenté à nouveau avec chaque nouvelle présentation du contenu du répertoire dans lequel ledit lien a été rangé ; cette présentation a lieu même si ledit lien a été supprimé dans le répertoire qui en était la source) – ceci par opposition aux liens en mode Suggéré qui ne sont plus présentés dès le moment où il sont supprimés dans le répertoire source ; cette présentation a lieu même si le répertoire qui en était la source n'est plus sélectionné (a été décoché).

2. si le répertoire dans lequel ledit lien est rangé est publié, de publier ledit lien d'office (sauf instruction contraire de l'utilisateur) – ceci par opposition aux liens ajoutés en mode Suggéré qui eux ne seront pas publiés.

On notera ici que l'insertion d'un lien dans un répertoire (par exemple manuellement en tapant son adresse URL ou par glisser-déposer à partir d'un autre répertoire) entraîne d'office sa mise en mode Accepté (puisque l'utilisateur l'a inséré lui-même).

Une page visualisée par l'utilisateur dans le répertoire courant entraîne l'insertion du lien vers ladite page en mode Suggéré dans ledit répertoire. Il n'est donc pas stocké. Toutefois, dès qu'un lien ajouté en mode Accepté est associé à ce lien, ce dernier passe en mode Accepté et le système le stocke donc dans l'espace personnel de l'utilisateur.

Section 10 - Liens en mode gelé

Le passage en mode « gelé » signifie que :

- non seulement l'utilisateur a validé le lien ajouté, mais de plus

- la version courante de la page (vers laquelle pointe ledit lien ajouté au moment où la transition au mode gelé est effectuée) est stockée dans le système afin de garantir à

l'utilisateur que ledit lien ajouté (aussi longtemps qu'il n'est pas supprimé) dirigera toujours l'utilisateur sur cette même version de la page dans le futur.

Plus précisément, la version courante de la page en question est copiée dans un espace de stockage propre au répertoire courant de l'utilisateur, et le lien pointe maintenant vers cette version stockée.

En réalité, pour optimiser la place consommée, il est avantageux que le système ne stocke qu'une seule copie de chaque version différente de page, qui sera partagée par tous les utilisateurs et tous les répertoires possédant ce lien en mode Gelé. Cette optimisation est nécessaire car une même version de page peut être pointée par des liens ajoutés associés à des pages différentes :

- d'un même répertoire,
- de répertoires différents d'un même utilisateur, ou encore
- de répertoires d'utilisateurs différents,

et par conséquent une multitude de liens ajoutés gelés pointant sur une même page peuvent exister.

Lorsqu'un lien ajouté pointant sur une page est gelé, le système vérifie d'abord si la version courante de ladite page est déjà copiée, le cas échéant le système modifie ledit lien ajouté pour le faire pointer vers cette copie.

Pour déterminer si une copie de la version courante de la page existe déjà ou pas :

- le système mémorise dans une table, en association avec chaque lien (vers une page) pour laquelle une copie est stockée, la date et l'heure de la copie ;

- dans la mesure où une copie existe pour le lien en question et où la copie est récente, le système compare le contenu de la version courante avec celui de la copie, et si les contenus sont identiques, le système évite ainsi de créer une nouvelle copie ;

- alternativement, le système peut exploiter les renseignements fournis dans le document source de la page en question pour connaître sa version (et déterminer si elle est différente de la version telle que copiée).

Section 11 - Contenants et contenus

On va maintenant généraliser les concepts décrits dans les sections précédentes, en considérant des structures « Contenants » (« Containers » en terminologie anglo-saxone) contenant des éléments « Contenus » (« Contents » en terminologie anglo-saxone), à n'importe quel niveau dans la structure des informations gardées par l'utilisateur dans son espace de stockage personnel.

Les différents modes d'exploitation des liens ajoutés entre pages décrits par ailleurs dans le présent chapitre pourront, en de nombreuses instances, être étendus à l'exploitation de liens situés dans des contenants inclus directement dans la structure de pages et pointant vers des contenus susceptibles d'être incorporés à la page en question.

Les pages sont des documents, eux-mêmes structurées sous forme hiérarchique (par exemple : sections, paragraphes, etc.), et auxquels est associée une spécification de présentation à l'utilisateur. Dans la suite, on va considérer des documents dont le contenu est spécifié selon la notation standard XML (abréviation de l'expression anglo-saxonne Extended Markup Language), et dont la présentation est spécifiée selon le langage XSL (abréviation de l'expression anglo-saxonne XML Stylesheet Language) ou selon un langage équivalent. Ces technologies se prêtent parfaitement à une structuration hiérarchique des informations.

Toute la toile personnelle de l'utilisateur est vue comme une structure arborescente de contenus/contenants imbriqués, chaque contenu étant vu comme un document (ou fragment de document) au format XML auquel est optionnellement associée une

spécification de présentation au format XSL. A chaque niveau dans cette structure d'arbre, un contenu ne peut être affiché à l'écran de l'ordinateur de manière autonome que si un document XSL lui est associé. Dans le cas inverse, le Content ancêtre le plus proche ayant un XSL est affiché par défaut.

Ainsi par exemple, un répertoire est d'abord un contenu qui peut être présenté à l'utilisateur comme une page autonome, dans la mesure où un document XSL pour la présentation des répertoires lui est associé. Un répertoire a par ailleurs un et un seul contenant dont les contenus sont eux-mêmes des répertoires ou d'autres types de documents. Certains de ces documents peuvent contenir plusieurs contenants, qui eux-mêmes contiennent des contenus, et ainsi de suite.

Tous les moyens décrits jusqu'ici pour les répertoires et les pages sont applicables respectivement aux contenants et contenus.

Dans ce cas, notamment :

- les liens (vers des pages) sont alors des liens vers des contenus ; les liens Ajoutés pointeront également sur des contenus,
- et les Coefficients de Probabilité d'Acceptation introduits dans la section « Détection de Répertoires Proches » sont alors associés à des contenants plutôt qu'à des Répertoires.

En partant du fait que l'utilisateur peut copier (ou « importer ») un lien d'un répertoire à un autre, la présente section propose un système pour lui permettre d'importer n'importe quel contenu de n'importe quel contenant à un autre (à condition que le type dudit contenu ne viole pas les contraintes de type du contenant qui est censé le recevoir) à n'importe quel niveau dans la structure des informations de l'utilisateur.

De même, cette extension du système a pour objet de permettre à l'utilisateur de tirer (c'est-à-dire de créer par glisser-déposer) des liens ajoutés de n'importe quel contenu

à n'importe quel autre contenu. Ainsi, au lieu de ne pouvoir tirer des liens ajoutés qu'entres pages, entre répertoires et d'une manière mixte entre répertoires et pages, l'utilisateur pourra tirer un lien ajouté entre un élément dans une page et une autre page considérée dans sa globalité, par exemple.

Avantageusement, un lien ajouté (et notamment un lien ajouté inverse) pourra pointer sur un contenu enfoui dans une page. Dans le cas où le contenu vers lequel pointe le lien ajouté n'a pas de spécification de présentation (au format XSL) associé, le mécanisme mis en œuvre affichera le plus proche contenu parent (qui le contient) dans la hiérarchie des contenus.

On introduira en outre une troisième manipulation offerte à l'utilisateur : la « dérivation » de contenant, selon laquelle un premier contenant dérivé d'un deuxième contenant reçoit en mode Suggéré tous les contenus du deuxième contenant (ou éventuellement une sélection de ceux-ci) qui sont en mode Accepté (ou Gelé). Cette transmission de contenus se fait en permanence, en ce sens que même les contenus qui sont acceptés dans le futur dans le deuxième contenant sont suggéré dans le premier contenant dès leur acceptation dans le deuxième contenant.

Les documents que constituent les pages publiées dans les sites Internet doivent être restructurés pour permettre d'en distinguer les contenus et contenants qui les constituent.

Le texte ci-dessous est un exemple simple de document en langage XML :

```
<Root>
        exemple d'informations <img src="urlsource"/> de toutes sortes
        <A id="A-1" name="Anatole">
                <C name="Hubert" id="C-1"/>
                <C name="Edouard"/>
                <C name="Antoine"/>
```

```
            <C name="Raymond"/>
        </A>
        <B name="Henri"/>
        <B name="Claude" myfriend="C-1" myenemies="A-1 B-1">
            exemple informations texte exemple informations texte
            <D name="Dominique" myfriend="A-1"/>
        </B>
        <B name="Robert" id="B-1"//>
        <B name="Andre"/>
</Root>
```

Un exemple de feuille de style XSL, permettant d'obtenir le rendu, est présenté dans les dix paragraphes suivants.

Entête de la feuille de style :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
   <HTML>
    <HEAD />
    <BASEFONT FACE="ARIAL" SIZE="2">
     <xsl:apply-templates />
    </BASEFONT>
    </HTML>
   </xsl:template>
```

Les nœuds du code source pour lesquels aucun traitement particulier n'est prévu sont copiés dans la page HTML :

```
    <xsl:template match="*">
     <xsl:copy>
      <xsl:apply-templates select="@*|node()" />
```

```
  </xsl:copy>
 </xsl:template>
```

Les attributs du code source pour lesquels aucun traitement n'est prévu prennent les valeurs spécifiés dans le code source :

```
 <xsl:template match="@*">
  <xsl:attribute>
   <xsl:value-of />
  </xsl:attribute>
 </xsl:template>
```

Pour le tag Root, sont ajoutés un titre et un paragraphe :

```
 <xsl:template match="Root">
  <H1>FAMILLE/AMITIE/ETC...</H1>
   <p>Dans la rue, il y a plein de gens :</p>
  <xsl:apply-templates />
 </xsl:template>
```

La valeur de l'attribut name est inscrite en caractères gras.

```
 <xsl:template match="*[@name]">
  <li>
   <b>
    <xsl:value-of select="@name" />
   </b>
```

Si le nœud possède un attribut id, la valeur de l'attribut est placée dans le rendu entre parenthèses :

```
    <xsl:if test="@id">
     (id=
     <xsl:value-of select="@id" />
     )
```

```
</xsl:if>
```

Si le nœud possède un attribut myfriend, la valeur de l'attribut est placée dans le rendu entre parenthèses :

```
<xsl:if test="@myfriend">
 (ami de
 <xsl:value-of select="@myfriend" />
 )
</xsl:if>
```

Si le nœud possède un attribut myenemies, la valeur de l'attribut est placée dans le rendu entre parenthèses :

```
<xsl:if test="@myenemies">
 (ennemi de
  <xsl:value-of select="@myenemies" />
 )
</xsl:if>
<xsl:choose>
```

Si un nœud possédant un attribut name possède des sous-nœuds avec des attributs name, ces sous-nœuds sont présentés dans une liste précédé du label "Enfants:" :

```
 <xsl:when test="*[@name]">
  <blockquote>
   Enfants :
   <ol>
    <xsl:for-each select="*[@name]">
     <xsl:apply-templates select="." />
    </xsl:for-each>
   </ol>
  </blockquote>
 </xsl:when>
```

Si un nœud possédant un attribut name ne possède pas des sous-nœuds avec des attributs name, le label "Aucun enfant" est affiché :

```
            <xsl:otherwise>- Aucun enfant</xsl:otherwise>
          </xsl:choose>
        </li>
      </xsl:template>
</xsl:stylesheet>
```

La structure de données XML est ici décomposée en un ensemble de contenants et de contenus. Ces derniers contiennent les éléments de la structure XML initiale. Dans un but de classification, les contenants peuvent optionnellement être associés à des catégories, de telle sorte que les contenus qu'ils possèdent soient classés dans cette catégorie.

Sur la figure 68, dans la structure arborescente d'un document, on voit apparaître des contenants (illustrés en pointillés).

La figure 69 illustre le fait qu'une telle structure permet de prendre des éléments d'un contenant pour les placer dans d'autres par le procédé importation.

On décrira également un procédé de dérivation permettant de placer un contenant dans un autre, dans le but de profiter du contenu du contenant d'origine, comme schématisé par la figure 70.

La figure 71 illustre le fait que ces manipulations peuvent également être effectuées d'un document à l'autre.

On cherche à pouvoir, à partir de pages Internet, restructurer les informations qui s'y trouvent afin de permettre à l'utilisateur de se composer ses propres pages, et ceci à partir de contenants et de contenus qu'il trouve au cours de sa navigation. Cela

implique de définir une macrostructure de contenants/contenus, et de dissocier les contenus, qui pourront ainsi être déplacés ou reproduits d'une page à l'autre.

Sur cette base, les données XML d'un document de la Toile seront décomposées pour adopter la structure suivante :

- un arbre de structure définissant l'imbrication des différents contenants d'information, chacun portant les références de ses contenus ;
- une suite de contenus, regroupant les éléments référencés dans les contenants.

Etapes de la transformation

L'administrateur d'une page d'un site dispose d'un code XML initial (on reprendra l'exemple déjà présenté plus haut), qu'il désire faire partager aux utilisateurs du système .

A. Création de la macrostructure

Pour que le code « source » puisse être restructuré et stocké en évitant les redondances, l'utilisateur spécifie quels éléments sont des contenus et à quel contenant chaque contenu appartient. Ceci permet de constituer ce qu'on appelle ici la macrostructure.

Les contenants sont spécifiés au moyen d'attributs. Utiliser des attributs plutôt que des « balises » (« tags » en terminologie anglo-saxonne) offre l'avantage de ne pas modifier la structure et ainsi, notamment, de ne pas remettre en cause l'application de feuilles de style XSL. La restructuration est ainsi « non intrusive ». On va maintenant présenter l'application d'une méthode de spécification de contenus et contenants (en se servant de l'exemple déjà présenté plus haut) au moyen d'attributs :

<Root CONTAINERNUMBER="1" CATEGORY="url1">

```
exemple d'informations <img src="urlsource"/> de toutes sortes
<A id="A-1" name="Anatole">
<C name="Hubert"    CONTAINERNUMBER="1"" CATEGORY="url2" id="C-1"
/>
<C name="Edouard" CONTAINERNUMBER="2" CATEGORY="url3"/>
<C name="Antoine"   CONTAINERNUMBER="2" CATEGORY="url3"/>
<C name="Raymond" CONTAINERNUMBER="2" CATEGORY="url3"/>
</A>
<B name="Henri" />
<B name="Claude" CONTAINERNUMBER="1" CATEGORY="url4"
   myfriend="C-1" myenemies="A-1 B-1">
exemple informations texte exemple informations texte
<D name="Dominique" myfriend="A-1"/>
</B>
<B name="Robert" CONTAINERNUMBER="1" CATEGORY="url4" id="B-1"/>
<B name="Andre" CONTAINERNUMBER="1" CATEGORY="url4" />
</Root>
```

On notera que :

- « Hubert » est dans un container (CONTAINERNUMBER= "1") différent de celui d' « Antoine », « Edouard » et « Raymond » (CONTAINERNUMBER= "2").
- les attributs « myfriend » et « myenemies » sont des références aux nœuds de l'arbre, car ils ont été spécifiés comme tels (type « idref ») par le schéma associé au code source en langage XML.

Remarques :

1- La macrostructure permet de dissocier les éléments, de la structure dans laquelle ils résident. Ainsi, les éléments sont stockés de manière unique sur le serveur quel que soit le nombre de fois où ils ont été reproduits.

2- Par défaut, sans aucune intervention manuelle, la macrostructure pourrait être automatiquement calquée sur la structure du document. La granularité du partage (granularité des éléments qui peuvent être répliqués) serait alors celle des éléments du document. Mais le système serait alors lourd à manipuler pour l'utilisateur, la macrostructure serait volumineuse et les performances s'en ressentiraient. L'approche que nous présentons consiste à permettre de spécifier la macrostructure explicitement dans les documents, c'est-à-dire de choisir la granularité des contenus qui peuvent être partagés. Ainsi, lors de la spécification d'un contenu, celui-ci est, par la même occasion, localisé dans un contenant qui peut regrouper plusieurs contenus. Comme les contenus auraient pu être calqués sur la structure des éléments du document, les contenants auraient pu être calqués sur la structure des contenus (tous les contenus enfants d'un même parent auraient alors été mis dans un contenant). On a préféré ici spécifier explicitement quels contenus sont situés dans quels contenants.

B. Code décomposé

A partir des indications figurant dans le nouveau code source, est construite la structure décomposée, constituée d'une part par l'arbre de structure et d'autre part la suite de contenus.

Dans notre exemple, l'arbre de structure est la suivante :

<CONTAINER SRC="...?id=12" CATEGORY="url1">
<CONTENTREF id="ContentRef-1" SRC="...?id= Content-42" NAME="Root">

<POSITION_CONTAINER N="1">

(Container où se trouve Hubert)
<CONTAINER SRC="...?id=13" CATEGORY="url2">
<CONTENTREF id="ContentRef-2" SRC="...?id= Content-43" NAME="C">

```
</CONTAINER>

(Container où se trouvent Antoine, Edouard et Raymond)
<CONTAINER SRC="...?id=14" CATEGORY="url3">
<CONTENTREF id="ContentRef-3" SRC="...?id= Content-44" NAME="C">
<CONTENTREF id="ContentRef-4" SRC="...?id= Content-45" NAME="C">
<CONTENTREF id="ContentRef-5" SRC="...?id= Content-46" NAME="C">
</CONTAINER>

</POSITION_CONTAINER >

<POSITION_CONTAINER N="2">

<CONTAINER SRC="...?id=15" CATEGORY="url4">

<CONTENTREF id="ContentRef-6" SRC="...?id= Content-47" NAME="B">

<POSITION_ELTREFS N="1">
<ELTREFS SRC="... ?id=ContentRef-2"/>
</POSITION_ELTREFS >

<POSITION_ELTREFS N="2">
<ELTREFS SRC="... ?id=ContentRef-1"/>
<ELTREFS SRC="... ?id=ContentRef-7"/>
</POSITION_ELTREFS >

<POSITION_ELTREFS N="3">
< ELTREFS SRC="... ?id=ContentRef-1"/>
</POSITION_ELTREFS >

</ CONTENTREF>
```

```
<CONTENTREF id="ContentRef-7" SRC="...?id= Content-48" NAME="B"/>
<CONTENTREF id="ContentRef-8" SRC="...?id= Content-49" NAME="B"/>
</CONTAINER>
</POSITION_CONTAINER >
</CONTENTREF>
</CONTAINER>
```

La suite de contenu est la suivante :

```
<CONTENT id=" Content-42" CATEGORY="url1">
    <Root>
        exemple d'informations <img src="urlsource"/> de toutes sortes
        <A name="Anatole" id="A-1">
            <POSITION_CONTAINER N="1"/>
        </A>
        <B name="Henri" />
        <POSITION_CONTAINER N="2"/>
    </Root>
</CONTENT>

<CONTENT id="Content-43" CATEGORY="url2">
    <C name="Hubert" id=C-1/>
</CONTENT>

<CONTENT id=" Content-44" CATEGORY="url3">
    <C name="Edouard"/>
</CONTENT>

<CONTENT id=" Content-45" CATEGORY="url3">
    <C name="Antoine"/>
```

```
</CONTENT>

<CONTENT id=" Content-46" CATEGORY="url3">
        <C name="Raymond"/>
</CONTENT>

<CONTENT id=" Content-47" CATEGORY="url4">
        <B name="Claude" myfriend="C-1" myenemies="A-1 B-1">
                <POSITION_ELTREFS N="1" ATTRIB="myfriend"/>
                <POSITION_ELTREFS N="2" ATTRIB="myenemies"/>
                exemple informations texte exemple informations texte
                <D name="Dominique" myfriend="A-1">
                        <POSITION_ELTREFS N="3" ATTRIB="myfriend"/>
                </D>
        <B/>
</CONTENT>

<CONTENT id="48" CATEGORY="url4">
        <B name="Robert" id="B-1"/>
</CONTENT>

<CONTENT id="49" CATEGORY="url4">
        <B name="Andre"/>
</CONTENT>
```

Comme on peut le voir, toutes les données d'arbre de départ sont séparées dans la suite de contenus. L'arbre de structure représente l'imbrication des contenants entre eux.

L'imbrication entre « Root » et « Henri » ne figure pas dans l'arbre de structure car « Root » et « Henry » sont dans le même contenu. Par contre, l'imbrication entre

« Anatole » et « Hubert », « Edouard », « Antoine » et « Raymond » est reflétée dans l'arbre de structure, car « Hubert », « Edouard », « Antoine » et « Raymond » sont regroupés dans deux contenants différents sous « Anatole ».

De plus, on constate que les contenants peuvent être disposés de façon bien précise dans les contenus. On parlera de positionnement. Les contenus dans lesquels se trouvent des contenants portent une balise indiquant où les contenants se trouvent (au moyen d'un numéro POSITION_CONTAINER N). Les positions dans les contenus sont utilisés dans l'arbre de structure. Chaque contenant est en effet imbriqué sous une balise POSITION_CONTAINER ayant un numéro. Ce numéro correspond à l'indication « N » figurant dans les contenus. Ceci est illustré sur la figure 72.

Lorsque certaines balises possèdent des références à d'autres balises, cela est matérialisé dans l'arbre de structure et dans la suite de contenus au moyen de la balise POSITION_ELTREFS. La même approche que pour le positionnement des contenants est utilisé. La position de références à des éléments est matérialisée dans un contenu à l'aide d'un numéro POSITION_ELTREFS N, que l'on retrouve dans l'arbre de structure, dans le champ « CONTENTREF » correspondant. Cette indirection permet de retrouver, dans la structure arborescente de contenants, le contenu où se trouve l'élément référencé. Ceci est illustré à la figure 73.

Détail de la macrostructure décomposée

a. Arbre de structure

1. Balise « CONTAINER »

Cette balise représente un contenant au sein de l'arbre de structure. Un contenant est identifié par une adresse URL dénommée SRC.

Cette adresse URL est composée du nom de domaine où le contenant est archivé, et de l'identifiant local de ce contenant pour le domaine. Si plusieurs serveurs sont associés à un nom de domaine, alors ce nom de serveur est spécifié dans l'adresse URL entre le nom de domaine et l'identifiant local :

SRC : DomainName/ServerName?ID

Cette balise comporte également un attribut indiquant à quelle catégorie est associé le contenant. Cette catégorie est spécifiée au moyen d'une adresse URL.

Un contenant est composé de plusieurs balises CONTENTREF (voir définition précise ci-après) qui renvoient au contenu dans la succession de contenus. De plus, un contenant peut être dans un contenu à une position donnée, selon la balise POSITION_CONTAINER (définie ci-après) désignant la position à laquelle il se trouve.

Par exemple :

<CONTAINER SRC=«... ?id=12» CATEGORY=«URL»>

2. Balise « CONTENTREF »

Cette balise constitue une référence à un élément donné CONTENT de la succession de contenus. Il est identifié par un identificateur entier « ID », et porte un attribut SRC qui est l'identifiant unique du contenu CONTENT référencé. La même syntaxe d'adresse URL que pour les contenants est adoptée. Les balises CONTENTREF comportent également un nom NAME :

<CONTENTREF ID= «432» SRC=«... ?43» NAME=«A»/>

3. Balise « POSITION_CONTAINER »

Cette balise permet de matérialiser les différentes positions où on peut trouver des contenants dans un contenu. Comme le montre l'exemple ci-après, une balise CONTENTREF peut contenir plusieurs balises POSITION_CONTAINER, c'est-à-dire que des contenants peuvent être disposés à plusieurs endroits du contenu en question. De plus, à une position donnée, il est possible de trouver plusieurs contenants, comme dans l'exemple suivant :

```
< CONTENTREF ID=26 SRC=«... ?ID=44» NAME=«B»>
<POSITION_CONTAINER N=1>
<CONTAINER SRC=«... ?ID=13» CATEGORY=«URL1»>
< CONTENTREF ID=27 SRC=«... ?ID=45» NAME=«B1»/>
</CONTAINER>
<CONTAINER SRC=«... ?ID=14» CATEGORY=«URL2»>
< CONTENTREF ID=28 SRC=«... ?ID=46» NAME=«B2»/>
</CONTAINER>
</POSITION_CONTAINER>
<POSITION_CONTAINER N=2>
<CONTAINER SRC=«... ?ID=13» CATEGORY=«URL1»>
<CONTENTREF ID=29 SRC=«... ?ID=47» NAME=«B3»/>
</CONTAINER>
</POSITION_CONTAINER>
</ CONTENTREF>
```

b. Suite de contenus

1. Balise « CONTENT »

Cette balise représente un contenu. Un contenu peut contenir une ou plusieurs balises du code source XML d'origine. Les contenus sont identifiés par un identifiant

unique, et portent l'adresse URL de la catégorie à laquelle ils appartiennent, comme dans l'exemple suivant :

<CONTENT ID=«43» CATEGORY=«URL»>
<A>
<C/>
</A>
</CONTENT>

Il se peut que certains cas conduisent à la création de contenus :

(i) soit des contenus externes, à savoir un contenu qui pointe sur un autre contenu d'un hôte différent (de catégorie externe), par exemple :

<CONTENT ID=«124» CATEGORY=«URL2(externe)»>
<A>
<C/>
</A>
</CONTENT>

(ii) soit des contenus indirects, à savoir un contenu qui pointe sur un autre contenu du même hôte (mais de catégorie différente), par exemple :

<CONTENT ID=«44» CATEGORY=«URL3» INDIRECT=«43» >
</CONTENT>

2. Balise « POSITION_CONTAINER »

Cette balise indique la position où un contenant peut être inséré dans un contenu, par exemple :

< POSITION_CONTAINER N=«1»/>

3. Balise « POSITION_ELTREFS »

Cette balise permet de matérialiser, dans le contenu, les références que certaines balises portent sur d'autres. Comme les balises « POSITION_CONTAINER », ils portent un attribut entier « N » indiquant leurs positions dans le contenu. Ils possèdent également un attribut « ATTRIB » qui contient le nom de l'attribut utilisé par l'auteur du code source XML pour définir la référence. Par exemple :

<POSITION_ELTREFS N=«1» ATTRIB=«myfriends»/>

C. Recomposition des pages de la Toile

La figure 74 illustre le fait que l'internaute dispose d'un accès supplémentaire à l'information. Il peut y accéder de façon classique en se connectant sur le site source de la Toile, et il peut également y accéder via le système, qui lui permet d'interagir avec les contenants et les contenus, c'est-à-dire les organiser selon sa volonté.

Le système décompose ce site en contenants/contenus (ce site doit impérativement avoir été mis par son administrateur sous la forme déjà décrite auparavant, à savoir la forme dans laquelle l'administrateur a notamment indiqué où se trouvent les contenus et contenants).

Afin d'être présentées à l'internaute, les informations en sortie du système sont reconstituées et transformées à l'aide de la feuille de style XSL associée au code XML d'origine. Pour permettre la manipulation de « poignées » (qui permettent de désigner graphiquement des contenus, la feuille de style XSL ajoute à la volée, dans la spécification XSL d'origine, du code de présentation. Ce procédé est schématisé sur la figure 75.

D. Code XML reconstitué

Afin de pouvoir être présenté à l'internaute, le code XML décomposé est reconstitué. On peut ainsi lui appliquer une feuille de style XSL. Lors de la reconstitution, il est fait en sorte que les contenus et les contenants soient matérialisés pour que l'utilisateur puisse les manipuler.

A cet égard, comme on va le voir par la suite, l'internaute peut sélectionner un contenu ou un contenant au moyen de « poignées » pour les déplacer où les mettre dans un autre document.

Pour l'exemple traité jusqu'ici le code reconstitué est le suivant :

<Root CONTAINERNUMBER="1" CATEGORY="url1">

Exemple d'informations de toutes sortes

<A id="A-1" name="Anatole">

<C name="Hubert" CONTAINERNUMBER="1"
CONTAINERID="13"
CONTENTID="43"
CATEGORY="url2" id="C-1" />

<C name="Edouard" CONTAINERNUMBER="2"
CONTAINERID="14"
CONTENTID="44"
CATEGORY="url3"/>

<C name="Antoine" CONTAINERNUMBER="2"
CONTAINERID="14"

```
CONTENTID="45"
CATEGORY="url3"/>

<C name="Raymond"  CONTAINERNUMBER="2"
CONTAINERID="14"
CONTENTID="46"
CATEGORY="url3"/>

</A>
<B name="Henri" />

<B name="Claude"    CONTAINERNUMBER="1"
CONTAINERID="86"
CONTENTID="47"
CATEGORY="url4"
myfriend="C-1" myenemies="A-1 B-1">

Exemple informations texte exemple informations texte

<D name="Dominique" myfriend="A-1"/>

</B>

<B name="Robert"    CONTAINERNUMBER="1"
CONTAINERID="87"
CONTENTID="48"
CATEGORY="url4" id="B-1"/>

<B name="Andre"    CONTAINERNUMBER="1"
CONTAINERID="88"
CONTENTID="49"
```

CATEGORY="url4" />

</Root>

Exemple de présentation

A partir du code XML reconstitué, une page HTML de présentation est construite au moyen d'une feuille de style XSL. Dans le cadre de l'exemple présenté en introduction de cette section, la même feuille XSL est reprise, à ceci près qu'on lui a ajouté des « éléments de dérivation » (c'est-à-dire les poignées) permettant d'interagir avec le contenu de la page.

Dans la suite, on va décrire précisément en quoi consistent ces éléments de dérivation, puis comment se passent les actions d'importation/dérivation et leur conséquence : la suggestion.

A. Insertion de poignées d'importation/dérivation

On définit à l'aide d'un modèle (« Template » en terminologie anglo-saxonne) la transformation en langage HTML des informations de la structure qui permettent la dérivation, l'importation et l'association de liens ajoutés. Les contenants ne peuvent pas être représentés en tant que tels, car ils ne correspondent pas forcément à une « enveloppe » (à savoir à un objet graphique unique) vue graphiquement dans la page HTML résultante. C'est pourquoi cette transformation donne un élément graphique (élément de dérivation) pour chaque nœud de type contenu. Cet élément de dérivation permet la gestion du glisser-déposer à la fois des contenants et des contenus. En effet, un composant « BEHAVIOR » (voir l'Annexe) attaché à l'élément communique alors avec le système.

Le modèle précité est ajouté en fin de la page XSL afin de la surcharger efficacement et pour que les éléments de dérivation soient effectivement affichés.

L'allure de ce modèle peut par exemple être la suivante :

```
<xsl:template match="*[@CONTAINERID>0]">
<xsl:if test="/*[@HANDLEMODE='1']">
<xsl:element name="DIV">
<xsl:attribute name="STYLE">
BEHAVIOR :url(.../system.dragndrop.htc) ;
</xsl:attribute>
<xsl:eval>this.nodeName</xsl:eval>
<xsl:attribute name="containerid">
<xsl:value-of select="@CONTAINERID"/>
</xsl:attribute>
<xsl:attribute name="contentid">
<xsl:value-of select="@CONTENTID"/>
</xsl:attribute>
</xsl:element>
</xsl:if>
<xsl:eval>
this.setAttribute("CONTAINERID",0)
</xsl:eval>
<xsl:apply-templates select="." />
</xsl:template>
```

L'élément DIV (introduit en 3e ligne) est un élément de dérivation auquel est associé le composant « BEHAVIOR :url(.../system.dragndrop.htc) », dont l'allure est présentée en Annexe. Ce composant permet une gestion du « glisser-déposer » des éléments pour la dérivation de contenants qu'il communiquera au système.

B. Réception de l'événement « ONDERIVE »

Lorsque le composant BEHAVIOR déclenche l'événement ONDERIVE, par l'instruction « fire("ONDERIVE",oEvent) » qui se trouve dans la fonction « FinishDrag » (la variable « oEvent » étant un objet événement dans lequel sont disponibles les informations (identifiants des éléments source et cible) qui permettent la dérivation et l'importation), le système est alors averti et réalise les actions suivantes :

1- il demande si l'utilisateur désire importer le contenu ou dériver le contenant ;.

2- il envoie la requête d'importation/dérivation ;

3- il vérifie si la page d'affichage est capable de prendre en compte la nouvelle source ;

4- il déclenche le rafraîchissement de l'affichage, de la façon suivante :

a. si la page est figée, reconstruction et traitement global de la source ;

b. si la page est dynamique, simple notification d'insertion de données.

Dans ce cas, la page reçoit la notification.

Cet événement fournit un pointeur sur un objet qui permet au script de la page de transformer cette nouvelle source (à l'aide, par exemple, de la feuille XSL qui l'a initialisée) et d'insérer le code HTML résultant dans la page.

C. Manipulation de la macrostructure

L'élaboration de la macrostructure de contenants/contenus permet :
- de placer de nouveaux contenus dans un contenant ; on parle alors d'importation de contenus ;
- de reproduire un contenant au sein d'autres contenus ; on parle alors de dérivation de contenants.

Importation

Supposons que l'on soit dans la situation illustrée dans la figure 76: les contenants 1 et 2 (« Container 1 » et « Container 2 ») sont rattachés à la catégorie 1 (symbolisé par la flèche grise). Dans le contenant 1 figure une référence au contenu 1.1 (« Content 1.1 »), et dans le contenant 2 figure une référence au contenu 1.2 (« Content 1.2 ») (ces références sont symbolisées par des flèches noires).

Importer le contenu 1.1 dans le contenant 2 revient à créer, dans ce contenant, une balise CONTENTREF qui pointe vers le contenu 1.1 dans la suite de contenus. Ceci se traduit par la figure 77.

Quand on importe un contenu d'une catégorie dans un contenant associé à une catégorie différente, on dit que l'importation provoque une re-catégorisation. Ceci se traduit par la création d'un contenu indirect, c'est à dire d'une balise similaire à une balise CONTENTREF mais qui s'en distingue par le fait qu'elle figure dans la suite de contenus, au sein d'une catégorie. Ceci est illustré sur la figure 78.

Dans le cas où les contenus importés possèdent des contenants, ces derniers sont dérivés suivant le mécanisme décrit ci-après.

Dérivation

Supposons que l'on dispose de l'imbrication contenant/contenu présentée sur la figure 79.

Dériver le contenant 1 revient à l'insérer dans un contenu à une certaine « position ». Pour ce faire, un nouveau contenant est créé dans l'arbre de structure, et pointe (c'est-à-dire possède une référence, matérialisée par une flèche grise sur le schéma de la figure 80a) vers le contenant que l'on veut dériver.

Sur le serveur, la structure de données a alors la forme suivante pour la partie dérivée :

- pour l'arbre de structure :

< CONTENTREF ID=26 SRC=«... ?ID=44» NAME=«Contenu»>
<POSITION_CONTAINER N=1>
<CONTAINER SRC=«... ?ID=126» SOURCESRC=«... ?ID=1»>
</CONTAINER>
<POSITION_CONTAINER>
</ CONTENTREF>

- pour la suite de contenus :

<CONTENT ID=«44» CATEGORY=«URL»>
<Contenu>
<POSITION_CONTAINER N=«1»/>
</Contenu>

</CONTENT>

Comme on peut le voir sur la figure 80a, le contenant 126 ne référence aucun contenu. Il a juste une référence sur le contenant source, à savoir celui à partir duquel on a effectué la dérivation. Le chargement des données dans le contenant dérivé se fait donc via le contenant source : il s'agit ici de « suggestion ». Cela conduit, sur le poste client, à la disposition illustrée sur la figure 80b.

Les sous-contenants sont dérivés exactement de la même manière que le premier contenant : un nouveau contenant est créé et pointe vers le sous-contenant source. La profondeur à laquelle on charge les éléments dans le contenant dérivé est fixée par le mécanisme de chargement progressif (voir plus loin chapitre V). Un tel mécanisme est nécessaire ici car, sinon, il pourrait se présenter des cas de boucles infinies dans le chargement. En effet, il est possible de dériver des contenants dans eux-mêmes, comme le montre la figure 81, et de créer ainsi une structure virtuellement infinie. Sans ce mécanisme, charger un contenant provoquerait alors une boucle infinie dans le cadre d'une structure réellement infinie.

Cas de la superposition de contenants

Comme on l'a déjà évoqué, les contenants sont insérés en des positions bien précises des contenus. Mais rien n'empêche d'insérer plusieurs contenants à une même position. On parle alors de superposition de contenants.

Partons de la structure présentée dans la figure 82 (en notant que les éléments situés dans un cadre en pointillés sont considérés comme étant dans un contenant).

Cela se traduit ainsi :

- pour l'arbre de structure :

<CONTENTREF ID=54 SRC=«... ?ID=42» NAME=«Root»>
<POSITION_CONTAINER N=1>
<CONTAINER SRC=«... ?ID=12» CATEGORY=«URL»>
<CONTENTREF ID=55 SRC=«... ?ID=43» NAME=«A»>
<POSITION_CONTAINER N=1>
<CONTAINER SRC=«... ?ID=13» CATEGORY=«URL»>
<CONTENTREF ID=56 SRC=«... ?ID=44» NAME=«C»/>
</CONTAINER>
</POSITION_CONTAINER>
</CONTENTREF>
<CONTENTREF ID=57 SRC=«... ?ID=45» NAME=«B»>
<POSITION_CONTAINER N=1>
<CONTAINER IDCONTAINER=«14»>
<CONTENTREF ID=58 SRC=«... ?ID=46» NAME=«D»/>
</CONTAINER>
</POSITION_CONTAINER>
</ CONTENTREF>
</CONTAINER>
</POSITION_CONTAINER>
</CONTENTREF>

- pour la suite de contenus :

<CONTENT ID=«42» CATEGORY=«URL»>
<Root>
<POSITION_CONTAINER N=«1»/>
</Root>
</CONTENT>

<CONTENT ID=«43» CATEGORY=«URL»>
<A>

<POSITION_CONTAINER N=«1»/>
</A>
</CONTENT>

<CONTENT ID=«44» CATEGORY=«URL»>
<B>
<POSITION_CONTAINER N=«1»/>
</B>
</CONTENT>

<CONTENT ID=«45» CATEGORY=«URL»>
<C/>
</CONTENT>

<CONTENT ID=«46» CATEGORY=«URL»>
<D/>
</CONTENT>

Supposons maintenant que l'on dérive le contenant 13 (celui qui contient le contenu C) dans le contenu 44 (qui correspond au contenu B) à la même position que le contenant 14 (qui contient le contenu D). La balise correspondant au contenu 44 dans la suite de contenus ne sera pas modifiée. En effet, le contenu conserve une seule position pour les contenants, même si deux contenants sont superposés sur cette position. Cette superposition se manifeste au niveau de l'arbre de structure par le fait qu'un nouveau contenant est ajouté sous la balise POSITION N=1, comme indiqué ci-dessous :

<CONTENTREF ID=32 SRC=«... ?ID=42» NAME=«Root»>
<POSITION_CONTAINER N=1>
<CONTAINER SRC=«... ?ID=12» CATEGORY=«URL»>
<CONTENTREF ID=33 SRC=«... ?ID=43» NAME=«A»>

<POSITION_CONTAINER N=1>
<CONTAINER SRC=«... ?ID=13» CATEGORY=«URL»>
<CONTENTREF ID=34 SRC=«... ?ID=44» NAME=«C»/>
</CONTAINER>
</POSITION_CONTAINER>
</CONTENTREF>
<CONTENTREF ID=35 SRC=«... ?ID=45» NAME=«B»>
<POSITION_CONTAINER N=1>
<CONTAINER SRC=«.. ?ID=14» CATEGORY=«URL»>
<CONTENTREF ID=36 SRC=«... ?ID=46» NAME=«D»/>
</CONTAINER>
<CONTAINER SRC=«15» CATEGORY=«URL»
SOURCESRC=«... ?ID=13»>
</CONTAINER>
</POSITION_CONTAINER>
</CONTENTREF>
</CONTAINER>
</POSITION_CONTAINER>
</CONTENTREF>

Mécanisme de Suggestion de nouveaux contenus (propagation dans la macrostructure)

Le fait de dériver un contenant implique qu'il est possible d'accéder aux contenus « acceptés » présents dans le contenant d'origine. Ces contenus sont en fait automatiquement « suggérés » dans les contenants dérivés au fur et à mesure de leur insertion.

C'est à l'utilisateur de préciser s'il accepte ou non les suggestions, comme décrit par ailleurs. Il est à noter que seuls les contenus acceptés pourront être dérivés à leur tour.

La suggestion peut ainsi se faire entre des utilisateurs différents (par exemple lorsqu'un utilisateur dérive un contenant d'un document dont il n'était pas l'auteur), mais également lorsqu'un même utilisateur dérive un contenant entre deux de ses propres documents.

Lorsqu'un utilisateur dérive un contenant, il crée un nouveau contenant portant la référence du contenant source, et les contenus du contenant source qui étaient en mode « Accepté » y sont chargés. Ces contenus sont initialement en mode « Suggéré », et l'utilisateur peut modifier leur mode.

Sur le plan de la base de données contenue dans le serveur, le contenant dérivé ne contient aucune information sur les contenus suggérés. Ces informations peuvent en effet être retrouvées via la référence sur le contenant source. En revanche, sur le poste client où ce contenant a été dérivé, une balise CONTENTREF est ajoutée pour chaque contenu chargé dans ce contenant, avec un attribut MODE=SUGGESTED.

Ce qui est suggéré peut être accepté ou refusé (ou encore laissé en mode suggéré). Cet éventuel changement de mode est répercuté par une mise à jour de la base de données sur le serveur, de la façon suivante :

- si le contenu est accepté : une balise CONTENTREF vers ce contenu est ajoutée dans la base de données dans le contenant dérivé, avec un attribut MODE=ACCEPTED.

- si le contenu est refusé : une balise CONTENTREF vers ce contenu est également ajoutée dans la base de données dans le contenant dérivé, avec un attribut MODE=REFUSED ; l'objectif dans ce cas est de noter que ce contenu ne doit pas être présenté à l'utilisateur à la prochaine connexion.

La mémorisation correspondant à ces différents cas est schématisée sur la figure 83.

Section 12 - Proposition de Répertoires Spécialistes et Voisins

On a vu dans la section précédente que, dans le cadre d'une page à laquelle l'utilisateur accède sur la Toile par des moyens standards, peuvent se trouver des contenants, et que dans chaque contenant peuvent se trouver des liens vers des contenus. Ces liens sont ceux choisis par l'administrateur du site d'où la page est issue comme devant être présentés par défaut.

Ces liens sont censés être en mode Suggéré. L'utilisateur, par l'intermédiaire du système, peut modifier leur mode, pour leur donner la valeur Accepté, Gelé ou Refusé, en accédant à cette même page avec le système.

Au lieu de présenter des mêmes liens à tous les utilisateurs, la fonctionnalité de proposition de répertoires Spécialistes (déjà décrite dans les sections « Publication de Liens Ajoutés » et « Publication de Répertoires ») permet de présenter plusieurs points de vue sur un sujet, selon une métaphore de « visite » des toiles personnelles de « Spécialistes ».

Avec ladite page, l'utilisateur peut aussi recevoir des liens ajoutés (qui y ont été associés par l'administrateur du site ou par les Spécialistes qu'il aura sélectionnés). Grâce à la fonctionnalité de proposition de répertoires Spécialistes, les liens ajoutés ne seront suggérés qu'en relation avec une « visite » de répertoire et ne surchargeront ainsi pas la page d'emblée.

Dans l'exemple que l'on prendra tout au long de cette section (voir figure 84a), pour une page sur le foie gras, des Spécialistes (ici des Chefs) différents présentent :

- associés à la page, des liens ajoutés sur des vins (Sauterne) et salades (Mesclun) différents ;

- à l'intérieur d'un contenant de la page, des liens sur différents types de foie gras (Canard, Oie).

L'utilisateur peut alors accepter certains liens ajoutés et/ou des liens situés à l'intérieur de contenants de la page, qui lui sont ainsi suggérés.

On notera que, dans ce cas, non seulement l'attribut « Mode » du lien ajouté mais aussi l'attribut Mode du lien dans le répertoire courant, pointant sur la page à laquelle est associé ce lien ajouté, est mis à jour à la valeur « Accepté ». L'acceptation remonte ainsi d'enfant à parent récursivement. Ainsi l'attribut Mode du lien pointant sur le répertoire courant, dans le répertoire qui le contient, le cas échéant, est aussi mis à jour, et ainsi de suite. Il en est de même pour les liens situés dans les contenants de la page.

Toutefois, en général, l'utilisateur n'accepte pas un lien ajouté (ou un lien dans un contenant) pointant sur une page P, sans au préalable cliquer sur lui pour visualiser la page P qu'il pointe. Ensuite, si la page P l'intéresse, il l'accepte directement, sans retourner accepter le lien ajouté d'où il est venu. En d'autres termes, au lieu d'accepter le lien ajouté (ou le lien dans un contenant) pointant sur P, il accepte la page P elle-même, c'est-à-dire le lien pointant sur ladite page P dans le répertoire courant qui l'accueille.

Un perfectionnement du système consiste alors à suivre le parcours de l'utilisateur qui clique sur un lien ajouté pour que, quand la page cible est acceptée, le système mette également à jour l'attribut Mode du lien ajouté par l'intermédiaire duquel il a accédé à cette page.

En acceptant un lien ajouté (ou un lien situé dans un contenant de page) l'utilisateur déclare son intérêt pour les contenus vers lesquels pointent ces Liens. Son « profil d'intérêt » se construit ainsi progressivement, et lorsqu'il atteint un seuil de

représentativité, le système peut lui proposer des répertoires proches dans la liste de répertoires « Voisins d'intérêt » (voir la liste « Copains » dans la figure 84a).

Dans le cas où, comme dans l'exemple de la figure 84a, la page courante comporte non seulement des liens ajoutés (Sauterne, Mesclun), mais aussi des liens dans au moins un contenant (Canard, Oie) se trouvant dans la page, le système sélectionne des répertoires qui sont proches:

- par rapport aux liens ajoutés acceptés associés à la page courante : le système sélectionne les répertoires dans lesquels se trouve la page courante ainsi qu'un grand nombre de liens ajoutés en commun, comme décrit dans la section « Détection de Répertoires Proches » ;

- par rapport aux liens acceptés) se trouvant dans chaque container se trouvant dans la page : le système sélectionne les répertoires contenant le plus grand nombre de liens en commun avec ledit contenant, comme décrit dans la section « Publication de Répertoires ».

On observera ici que le système peut exploiter non seulement les liens (liens ajoutés et liens dans les contenants) acceptés et gelés, mais aussi les liens refusés. Il est en effet intéressant de considérer ce que l'utilisateur n'a pas retenu dans les liens qui lui ont été proposé pour construire son profil.

Le système exploite ces mêmes données (liens ajoutés et liens dans les contenants) pour ajuster les répertoires Spécialistes proposés à l'utilisateur. Ceci est décrit dans les sections « Publication de Liens Ajoutés » et « Publication de Répertoires ».

En lui proposant un ensemble de répertoires proches, le système permet à l'utilisateur de visiter les toiles personnelles d'un ensemble d'autres utilisateurs, avec qui il peut par ailleurs se mettre en contact, comme déjà décrit dans le section « Détection de Répertoires Proches ».

L'utilisateur peut cliquer sur un lien (vers contenu) dans un contenant de la page, ou sur un lien ajouté pour accéder au contenu en question. Le système lui propose alors les répertoires « Spécialistes », « Voisins » et « Copains » pour ce nouveau contenu, et ainsi de suite.

Enfin, l'utilisateur peut sélectionner un ensemble de liens ajoutés, ou un ensemble de liens dans un contenant, pour demander au système de lui sélectionner des répertoires proches compte-tenu de cet ensemble. L'homme du métier saura facilement étendre les procédés décrits jusqu'ici pour mettre en œuvre cette fonctionnalité.

Section 13 - Attributs relatifs à l'acceptation

Lors de l'acceptation (passage au mode Accepté ou Gelé) d'un contenu, l'utilisateur peut spécifier s'il y est intéressé pour l'acheter, le vendre, et/ou pour d'autres intérêts qu'il y porte le cas échéant, en fonction du type de contenu.

Ces spécifications se font en donnant des valeurs à des *attributs* que le système prend en compte lors des sélections de répertoires (Spécialistes et Voisins) qu'il propose à l'utilisateur, ainsi que pour sélectionner le contenu de ces derniers le cas échéant (voir les sections « Détection de Répertoires Proches », « Suggestion de Liens Ajoutés par l'administrateur d'un site », « Publication de Répertoires » et « Proposition de Répertoires Spécialistes et Voisins »).

L'utilisateur qui accède à une page (page courante), peut spécifier des critères sur ces attributs pour restreindre l'ensemble des répertoires (Spécialistes et Voisins) qui lui sont proposés en relation avec ladite page courante.

Ainsi, si par exemple il est intéressé par l'achat du produit présenté dans la page courante (ou dans un ensemble de pages qu'il sélectionne), il pourra s'intéresser sélectivement aux répertoires d'utilisateurs :

- intéréssés à vendre ce produit (recherche de fournisseurs de ce produit),

- ou intéréssés comme lui à acheter ce produit : il pourra ainsi se joindre au groupe d'utilisateurs intéressés par cet achat et augmenter l'influence de ce groupe pour faire baisser le prix du produit chez un fournisseur, éventuellement à l'aide d'une procédure automatisée (d'aide à l'*achat groupé*) qui se déclenche quand ce critère est utilisé.

Dans le premier cas l'utilisateur recevra, à partir des répertoires de vendeurs ainsi sélectionnés, des liens en mode suggéré concernant des offres particulières pour le produit présenté dans la page courante ainsi que sur d'autres produits proposés par ces vendeurs.

Dans le deuxième cas, l'utilisateur bénéficiera des découvertes des autres utilisateurs (acheteurs potentiels) ayant des intérêts (et/ou goûts) proches.

Rapelons que les utilisateurs du système sont anonymes et peuvent y accéder en utilisant un pseudonyme.

On va maintenant décrire une mise en œuvre particulière de cet aspect de l'invention.

Seuls deux attributs sont mis en œuvre : « Demandeur » et « Offreur ».

Lors de l'acceptation d'un lien, en utilisant l'interface illustrée aux figures 55b et 66 (qui notamment lui permettent de changer le mode du lien de « suggéré » en « accepté » ou en « gelé »), l'utilisateur a deux « cases à cocher » à sa disposition, à savoir : « Demandeur » et « Offreur ».

Cette configuration est illustrée schématiquement à la figure 84b.

Il cochera « Demandeur » s'il se positionne comme demandeur, et sur « Offreur » s'il se positionne comme « offreur », vis à vis de la « chose » présentée dans la page courante. On observera que ces deux possibilités ne sont pas exclusives : il peut se positionner à la fois comme demandeur et comme offreur.

Le système mémorise la valeur de ces deux attributs pour chaque lien dans la toile personnelle de chaque utilisateur. Il peut ainsi en tenir compte dans le procédé de sélection des répertoires à proposer aux utilisateurs.

Pour actionner la sélection de répertoires proposés dans les deux premières listes (« spécialistes » et « voisins » ; voir la figure 55a) en fonction de ces deux attributs, l'utilisateur a deux autres cases à cocher « Demandeur » et « Offreur » à sa disposition.

Ces deux autres cases à cocher sont cochées par défaut.

En décochant « Demandeur », l'utilisateur spécifie qu'il ne s'intéresse pas à visiter les répertoires positionnés comme demandeur en regard de la chose présentée dans la page courante.

De même, en décochant « Offreur », l'utilisateur précise qu'il ne souhaite pas recevoir de liens à partir de répertoires positionnés comme offreur en ce qui concerne la chose présentée dans la page courante.

Ces attributs peuvent avoir une interprétation plus large que celle concernant l'achat et la vente. Par exemple, l'attribut Offreur peut être utilisé par l'utilisateur pour informer sur ses propres caractéristiques : l'utilisateur peut par exemple cocher l'attribut offreur sur une page présentant un acteur de cinéma qui lui ressemble, dans l'espoir de pouvoir se mettre en contact, au moyen du système (par exemple par vidéoconférence), avec une utilisatrice qui apprécie cet acteur tel qu'il est présenté dans ladite page.

Dans le cas de l'application d'*achat groupé* sus-mentionné, le système proposera à l'utilisateur des voisins qui sont « Demandeurs » du produit présenté dans la page courante visualisée. L'utilisateur pourra alors interagir avec ces utilisateurs par des moyens assistés par ordinateur, en communication synchrone (messagerie instantanée, etc) ou asynchrone (forum et courrier électronique augmentés de moyens semi-automatiques spéciaux pour faciliter la coordination d'achat groupé, notamment pour négocier les prix avec les Offreurs).

Section 14 - Pages Personnalisées

Les documents restructurés comme décrit dans la section précédente permettent de présenter, dans des contenants, des contenus sélectionnés en fonction des centres d'intérêts de l'utilisateur. On va maintenant décrire une façon dont peut être effectuée cette sélection.

Rappelons tout d'abord que tout contenu accepté par l'utilisateur dans un contenant adopte la catégorie de ce contenant (comme on l'a décrit notamment plus haut dans le présent chapitre et comme on y reviendra dans le chapitre III), en plus des catégories des autres contenants dans lesquels l'utilisateur a aussi accepté ce contenu le cas échéant.

Dans la mesure où le système connaît les catégories des contenants imbriqués dans un contenu qui fait l'objet d'une requête, il peut automatiquement enrichir la requête par des critères supplémentaires. En effet, le système est capable de déterminer que, pour un certain nombre de contenus de chacune desdites catégories, il existe d'autres catégories qui ont été attribuées par l'utilisateur à ces mêmes contenus. Le principe utilisé ici est d'exploiter ces autres catégories en les utilisant en tant que critères supplémentaires (ou plus exactement en tant que « préférences ») pour enrichir automatiquement la requête formée par l'utilisateur.

Par exemple, dans un contenu qui fait l'objet d'une requête, se trouve imbriqué un contenant de catégorie « guitare», et chez l'utilisateur, un certain nombre de contenus de cette catégorie possèdent également la catégorie « objet d'art ». On exploite donc le fait qu'a priori, dans ce contenant, l'utilisateur préférera recevoir (par le processus de suggestion décrit par ailleurs) des contenus qui possèdent non seulement la catégorie « guitare », mais également la catégorie « objet d'art ». Ainsi, pour le contenant de catégorie « guitare », la requête de l'utilisateur peut être enrichie par le critère supplémentaire : catégorie = « objet d'art ». On observera ici que, s'il n'existe aucun contenu appartenant à ces deux catégories simultanément, le système propose alors à l'utilisateur les contenus appartenant simplement à la catégorie « guitare ». Le critère supplémentaire sera ainsi pris en compte comme étant une « préférence ».

On comprend donc que les contenus sont sélectionnés par le système de manière « personnalisée », c'est-à-dire en tenant compte des centres d'intérêts potentiels de l'utilisateur pour les contenants en question.

Par ailleurs, les catégorisations effectuées par les utilisateurs peuvent « remonter » vers l'administrateur d'un site en tant que suggestions de catégorisation pour les contenus en question, de manière à modifier la structure XML de la page considérée si l'administrateur le juge opportun. Le procédé d'auto-épuration décrit en détail dans le chapitre suivant permettra d'automatiser, dans une certaine mesure, le filtrage des catégorisations considérées comme qualitativement mauvaises ou non fiables.

* * *

Chapitre III - Processus de suggestion généralisé (Figures 85 à 134)

Ce chapitre vise un cadre différent pour le processus de suggestion décrit plus haut dans les dernières sections du chapitre II ainsi que différentes extensions de ce processus. Ce processus va être décrit ci-dessous à des niveaux de fonctionnalités et de sophistication croissants. On présentera d'abord (Section 1) les structures et les

procédés fondamentaux de chaque niveau. La présentation de chaque niveau s'appuie sur les niveaux précédents. Dans la Section 2, on décrira les procédés annexes de filtrage par apprentissage de fiabilité de l'information. La Section 3 présente le principe de suggestions automatiques par rapprochement d'intérêts, et la section 4 présente un perfectionnement des mécanismes de suggestion automatique qui exploite l'avantage du graphe de dérivations entre utilistaurs. Ensuite sont présentés les méthodes de renforcement de l'anonymat des Utilisateurs et les modèles économiques possibles (Section 5). Enfin la Section 6 présente l'exploitation du mécanisme de dérivation/suggestion/acceptation décrit dans la première section pour la fidélisation des utilisateurs en exploitant leur fibre de collectionneur.

Section 1 - Structure et procédé fondamentaux

A. Bref rappel de la notation UML

Considérons la structure hiérarchique schématisée ci-dessous :

```
Contenant-1
        Contenus
                Contenu-1
                Contenu-2
                ...
        Sous-Contenants
                Contenant-11
                        Contenus
                        ...
                        Sous-Contenants
                        ...
                Contenant-12
                ...
Contenant 2
...
```

Cette structure hiérarchique est spécifiée sur la figure 85 des dessins qui est un diagramme de classe selon un standard connu de Modélisation Orienté-Objet, noté UML (voir en particulier « UML Reference Manual » – Rumbaugh, Booch, Jacobson, 1999).

Il s'agit d'une structure hiérarchique arborescente d'Eléments (« Elt »), chaque Elément étant un « Contenant » ou un « Contenu », les Contenants pouvant contenir des Eléments (ce sont leurs Sous-éléments « SousElt »), tandis que les Contenus ne peuvent pas contenir d'éléments.

Les Utilisateurs du système selon l'invention utilisent une telle structure arborescente d'Eléments.

B. forme de réalisation de base (niveau 1)

Tous les Utilisateurs du système utilisent la même structure de Contenants. Mais, dans de mêmes Contenants, chaque Utilisateur regroupe des Contenus qui peuvent être différents d'un Utilisateur à l'autre.

Tous les Contenus de tous les Utilisateurs - sauf les Contenus confidentiels (voir plus loin) - sont regroupés, dans leurs Contenants respectifs, dans une Base de Données Commune, notée « BD », accessible par tous les Utilisateurs .

On appelle ici « Profil » l'ensemble des Contenus d'un Utilisateur donné pour un Contenant donné.

Le diagramme de classe correspondant est illustré sur la figure 86.

Le système décrit ici offre pour les Utilisateurs une garantie d'anonymat : ainsi un Utilisateur ne peut pas accéder aux Profils d'un autre Utilisateur (les Profils sont

confidentiels), bien qu'il puisse accéder à tous les Contenus (non confidentiels), puisqu'il peut consulter l'ensemble de la Base de Données Commune.

En d'autres termes, la Base de Données Commune mémorise, dans chaque Contenant, l'union des Profils des Utilisateurs pour ce Contenant, sans permettre de distinguer les Profils eux-mêmes.

De là, un Utilisateur est incapable de déterminer dans quels Profils est référencé un Contenu donné auquel il accède dans la Base de Données Commune.

Cette organisation est schématisée sur la figure 87.

Lorsqu'un Contenu est modifié par un Utilisateur (bien entendu, à condition qu'il en ait le droit, par exemple s'il est le créateur de ce Contenu), les autres Utilisateurs, ayant ce Contenu dans leurs Profils respectifs, le voient mis à jour automatiquement dès qu'ils accèdent à la BD Commune pour la première fois postérieurement à ladite modification.

L'Utilisateur peut consulter la BD Commune. Il peut s'y abonner pour être automatiquement informé des ajouts de Contenus dans certains des Contenants, de façon sélective (par exemple au moyen de la puissance d'expression d'un langage classique de requêtes à une Base de Données). Le fait d'être informé automatiquement est ce qu'on appelle un « procédé de suggestion », dont on donnera davantage de détails dans la suite.

On va maintenant décrire un certain nombre de fonctionnalités possibles offertes aux Utilisateurs.

a) Ajout et Suppression de Contenu Direct

L'Utilisateur peut choisir d'ajouter dans un de ses Profils :

* des Contenus qu'il aura puisé dans le Contenant lié à ce Profil

* des Contenus qu'il aura puisé dans d'autres Contenants

* ou encore de nouveaux Contenus qui n'existaient dans aucun Contenant (qu'il aurait par exemple collecté par navigation sur le réseau Internet).

Quelle que soit sa provenance, si un Contenu ajouté dans un Profil n'existait pas déjà dans le Contenant lié à ce Profil, il y est ajouté. Les autres Utilisateurs peuvent alors en prendre connaissance.

On notera ici que, dans la pratique les Profils peuvent ne pas être directement composés de Contenus, mais de Références à ces derniers, ou encore d'un mélange de Contenus et de Références à des Contenus. La figure 88 illustre le diagramme de classe associé, qui comporte en particulier un lien entre Contenu et Référence à Contenu).

Dans un Profil, un Contenu peut posséder l'un parmi trois attributs, à savoir « Suggéré »», « Accepté » ou « Refusé ». Un Contenu « Suggéré » signifie que ce Contenu est reçu dans le Profil, sans pour l'instant avoir été accepté ou refusé par l'Utilisateur qui possède ce Profil. Les Contenus étant dans cet état ne sont pas stockés de manière permanente (c'est-à-dire qu'une Référence n'est pas encore créée dans la BD de Profils).

« Accepter » un Contenu signifie que le système crée de manière permanente une Référence à ce Contenu. Les Contenus Acceptés sont stockés dans le Profil de manière permanente. Il est ainsi à observer ici que l'acceptation au sens du présent chapitre correspond au mode « Gelé » décrit dans le chapitre II.

Un Contenu dont la présence dans un Profil est « Refusée » ne peut être supprimé directement dans le Contenant (c'est-à-dire dans la Base de Données Commune, où le Contenu est réellement stocké), car ce Contenu peut auparavant avoir aussi été Suggéré ou Accepté dans un autre Profil. Il ne pourra être réellement effectivement supprimé que lorsque plus aucun un autre Profil ne contiendra une Référence à ce

Contenu. En attendant, c'est l'attribut « Refusé » qui matérialise la suppression virtuelle voulue par un Utilisateur. La Référence ayant cet attribut est toutefois stockée de manière permanente et sert à éviter de suggérer à nouveau un Contenu qui a précédemment été « Refusé ».

b) Références à des Contenus d'autres Contenants ou à des Contenus externes (Contenus Indirects)

Dans une forme de réalisation particulière, le système peut permettre aux Utilisateurs d'insérer dans un Contenant (via leurs Profils respectifs), une Référence à un Contenu appartenant déjà à un autre Contenant. Ceci est préférable à l'insertion d'une copie de ce Contenu. Bien entendu, cette Référence ne pourra exister que tant que le Contenu référencé existe. Nous appelons cette référence « Contenu Indirect », et le Contenu Référencé est appelé « Contenu Direct ».

Avantageusement, à chaque nouvel accès à la Base de Données Commune, dans le cas où le Contenu Direct a été entre-temps modifié, l'Utilisateur voit, à travers le Contenu Indirect, ce Contenu dans sa nouvelle version. Le diagramme de classe UML associé est illustré sur la figure 89.

Supprimer un Contenu Direct implique au préalable de gérer les Contenus Indirects qui lui sont liés. Ils n'auront en effet plus lieu d'exister, mais comme expliqué précédemment, ils ne pourront être effectivement supprimés que quand plus aucun Profil ne les référencera. La suppression d'un Contenu Direct consistera donc en 3 types d'actions qui ne se sont pas nécessairement exécutées immédiatement :

(1) suppression, dans les Profils, des Références au Contenu Direct et aux Contenus Indirects qui lui son liés,

(2) suppression de ces Contenus Indirects,

(3) suppression du Contenu Direct lui-même.

Un Utilisateur, qui a référencé dans son Profil un Contenu Indirect pour lequel il est averti d'une suppression prochaine, pourrait vouloir créer une copie du Contenu Direct référencé à la place du Contenu Indirect, afin de ne pas perdre l'information. Le système peut être conçu pour lui en laisser la possibilité.

Les Contenus Indirects peuvent aussi constituer des références à des contenus externes au système, et notamment des liens vers des éléments trouvés sur l'Internet. Le système comporte alors des moyens de mise à jour (ou gestion des erreurs d'accès) en cas de suppression des contenus externes.

c) Popularité d'une catégorisation

Le fait d'insérer un Contenu dans un certain Contenant (via un Profil) revient à le « catégoriser ». Un Contenant est en effet censé regrouper des Contenus de même catégorie.

Différents Utilisateurs peuvent classer un même Contenu dans des Contenants différents, de manière directe (en en faisant une copie) ou au moyen d'une indirection (Contenu Indirect).

On appelle ici « Popularité » le nombre de fois qu'un Contenu se retrouve dans des Profils associés à un même Contenant, ou plus généralement une variable représentative de ce nombre. Dans le cas où l'Utilisateur hésite entre deux catégorisations possibles, il préférera en général le Contenant le plus populaire, afin de « parler le même langage » que le plus grand nombre d'Utilisateurs et de bénéficier ainsi de services de rapprochement de profils et d'autres avantages tels qu'on les décrira dans la suite.

L'avantage essentiel de cette forme de réalisation de base de l'invention peut être résumé par ce qui suit, en référence à la figure 90 :

- tout d'abord, un Utilisateur A, pour ses besoins propres de catégorisation et mémorisation d'informations (ceci est une incitation suffisante), va au fil du temps ajouter à son propre Profil A de nouveaux Contenus, dans des Contenants représentant des catégories partagées ;
- par le fait qu'il utilise ces Contenants, l'Utilisateur « parle le même langage » que les autres Utilisateurs et bénéficie ainsi d'avantages supplémentaires (tel que la suggestion automatique d'informations par rapprochements de profils, etc, décrits dans les sections suivantes) en échange du fait qu'il contribue à la BD Commune
- la BD Commune est ainsi alimentée de manière automatique et transparente, et un autre Utilisateur B peut la consulter ou être automatiquement et spontanément renseigné sur son enrichissement ;
- chaque Utilisateur peut être dans la situation A ou B ;
- le fait qu'un Utilisateur contribue à la Base de Données Commune ne le prive pas de sa privauté car il peut rester anonyme ;
- enfin le modèle économique décrit dans la dernière section permet de rétribuer ceux qui contribuent plus à la BD Commune qu'ils ne consultent, et de sanctionner l'inverse.

De façon encore plus résumée, l'avantage individuel de pouvoir organiser l'information (découverte par chacun), selon des catégories partagées est une incitation qui pousse chaque Utilisateur à contribuer à une BD qui est commune à tous les Utilisateurs.

C. Premier perfectionnement (niveau 2)

Selon ce premier perfectionnement, tous les Profils ne sont pas nécessairement confidentiels. Nous distinguons les Utilisateurs qui veulent publier certains de leurs Profils (ou un ou plusieurs sous-ensembles de leurs Profils) à l'intention des autres (ou de certains autres), par opposition aux Utilisateurs qui n'utilisent le système que pour leur organisation personnelle d'informations.

Plus exactement, nous distinguons les Profils (ou sous-ensembles de Profil) qui sont publiés à l'ensemble (ou à un sous-ensemble) des autres Utilisateurs et les nommons « Profil Non Confidentiel ».

On va maintenant décrire un certain nombre de fonctionnalités autorisées par ce perfectionnement.

a) Le concept de Dérivation

Les Utilisateurs peuvent « Dériver » un Profil Non Confidentiel, c'est-à-dire l'importer et l'utiliser comme étant leur propre Profil (Confidentiel ou pas) et en enlever ou y ajouter des Contenus.

On entend par là que les Contenus Ajoutés (ou Acceptés) dans le Profil source sont automatiquement Suggérés dans le Profil dérivé et que l'Utilisateur de ce dernier peut ainsi en Accepter tout ou partie.

Par « en enlever » on entend que l'Utilisateur peut Refuser certains Contenus Suggérés.

Finalement, l'Utilisateur peut aussi Ajouter :

* des Références à des Contenus, ou
* des Références à des Références se trouvant dans d'autres Profils.

Les nouveaux Contenus (Directs ou Indirects) sont alors ajoutés dans la BD Commune dans le Contenant lié au Profil dérivé, s'il n'y figuraient pas déjà.

Un diagramme de classe relatif à la structure d'un tel système de dérivation de Profils est illustré sur la figure 91 des dessins.

On observe qu'un Profil Dérivé et Non Confidentiel peut être dérivé à son tour par un autre Utilisateur. Les Références entre Profils peuvent donc se faire en cascade pour aboutir finalement aux Contenus dans la Base de Données Commune.

Optionnellement, un Profil Dérivé peut être fusionné avec un Profil déjà existant (que ce dernier soit lui-même Dérivé ou non) et adopter le Contenant de ce dernier. En résultat, un Profil peut ainsi être dérivé de plusieurs Profils.

b) Lien Profil-Contenu

Une même Référence peut en même temps être dans l'état Ajouté dans un Profil et dans l'état Refusé dans un autre Profil. On doit alors introduire un objet intermédiaire « Lien Profil-Contenu » (ou lien) pour spécifier cet état. Ainsi, une même Référence peut avoir un lien ajouté avec un Profil et un lien Suggéré avec un autre Profil.

L'état d'un lien Profil-Contenu peut être « Suggéré », « Refusé », « Ajouté » ou « Retiré ». Ceci sera précisé plus loin.

c) Archive

On peut également introduire une fonction d'« Archive de Profil », qui sert d'étape intermédiaire avant la suppression effective d'une Référence dans un Profil.

Une Référence R peut être Retirée dans un Profil P1 alors que dans un Profil P2 on y fait référence en mode Ajouté ou Suggéré. Dans ce cas, au moment où R est Retirée dans P1, elle est placée dans l'« Archive de Profil » liée à P1 et ne pourra être effectivement supprimée que lorsque plus aucun Profil n'y fera référence en mode Ajouté ou Suggéré.

Autrement dit, quand un lien Profil-Contenu passe à l'état « Retiré », si d'autres références existent sur la Référence en question, cette dernière se délie du Profil et se lie avec l'Archive de ce Profil, et si aucune référence n'existe sur elle, elle peut être supprimée effectivement (sous réserve de possibilité d'annulation de l'action par des méthodes traditionnelles).

Ainsi un attribut « Compteur de dérivation » peut être prévu pour mémoriser le nombre de Références Suggéré ou Ajouté sur la Référence en question. La Référence peut ainsi être supprimée de l'Archive quand la valeur du Compteur de Dérivation passe à zéro.

d) Procédé de Suggestion

La suggestion est le procédé de communication entre Profils. Par ce procédé, les Contenus se propagent :

* soit automatiquement
    -d'un Profil à ses Profils dérivés,
    - et inversément, des Profils dérivés aux Profils sources
* soit « manuellement », par le fait de soumettre un Contenu d'un Profil à un autre Profil.

On va détailler ci-dessous ces différents modes de propagation :

i) Propagation d'un Profil à ses Profils Dérivés

Les Références ajoutées dans un Profil d'un Utilisateur UB , lequel Profil est dérivé par un Utilisateur UA , sont par défaut automatiquement suggérés à UA, comme l'illustre la figure 92.

Le résultat de la suggestion se matérialise par un lien Suggéré Dérivé comme illustré sur la figure 93. Ce lien n'est pas rendu permanent, en ce sens qu'il n'a d'existence que sur le poste client.

Dans le cas où le possesseur du Profil A accepte la suggestion du Contenu pointé par R, il n'est plus besoin de suggérer à nouveau R à chaque connexion, car une autre Référence R' pointant sur R est créée, comme le montre la figure 94.

Le système utilisera alors le fait que l'état du lien Suggéré Dérivé est Consulté (le lien est stocké de manière permanente sur le serveur) pour ne pas suggérer à nouveau ce Contenu à la prochaine connexion de l'Utilisateur sur ce Profil.

ii) Propagation d'un Profil Dérivé à son (ou ses) Profil(s) source(s)

Les Références ajoutées dans un Profil Dérivé sont automatiquement suggérées au(x) propriétaire(s) du (des) Profil(s) d'origine. Bien entendu, ce (ou ces) derniers peu(ven)t les accepter ou les refuser.

Le schéma de la figure 95 illustre la communication de UB à Uc via le Profil de UA, Les Profils de UB et Uc ayant été dérivés du Profil de UA.

Le résultat de la suggestion automatique du Profil B au Profil A se matérialise par un lien Suggéré, comme illustré sur la figure 96, qui n'est pas stocké de manière permanente :

Dans le cas où le possesseur du Profil A accepte la Référence R, il n'est plus besoin de la suggérer à nouveau à chaque connexion, car une autre Référence R' sur R est créée. Pour le permettre, le lien Suggéré est rendu permanent. Ceci est illustré sur la figure 97.

iii) Référencement (Suggestion manuelle)

Par opposition aux suggestions automatiques évoquées ci-dessus, le Référencement est demandé explicitement par l'Utilisateur. Il consiste à « re-catégoriser » un Contenu dans un autre Contenant et entraîne la création d'un Contenu Indirect.

Un lien Profil-Contenu de Suggestion de Référencement a le même comportement que les liens Profil-Contenant de Suggestion automatique (déjà évoqués ci-dessus - voir aussi plus loin les spécifications des transitions d'état).

Le résultat de la suggestion de Référencement du Profil B au Profil A se matérialise par un lien Suggéré tel qu'illustré sur la figure 98, qui est rendu permanent (contrairement aux exemples précédents) :

On notera ici qu'en remontant le lien de Suggestion de Référencement, on peut retrouver le Profil dans lequel le Contenu en question a été référencé.

Dans le cas où le possesseur du Profil A accepte la Référence R, il n'est plus besoin de la suggérer à nouveau à chaque connexion, car une autre Référence R' pointant sur R est créée, comme illustré sur la figure 99.

Note sur les types de liens Profil-Contenu

Les Contenus (plus exactement des Références à des Contenus) situés dans un Profil Propre ou dans un Profil Dérivé ont des liens avec ces Profils qui peuvent être différents.

Ainsi les Profils Propres peuvent avoir :

* des liens Créés (liens avec de nouveaux Contenus Directs ou Indirects),
* des liens Dérivés (liens avec des Contenus dérivés d'autres Profils) ou
* des liens Suggérés (liens avec des Contenus qui lui ont été suggérés).

Les Profils Dérivés peuvent avoir quant à eux :

* des liens Créés (liens avec de nouveaux Contenus Directs ou Indirects),

* des liens Dérivés (liens avec des Contenus dérivés d'autres Profils)

* des liens Suggérés Dérivés (liens avec des Contenus appartenant au Profil source) ou

* des liens Suggérés (liens avec des Contenus qui lui ont été suggérés).

Le diagramme de classe illustrant ces différents types de liens est illustré sur la figure 100 des dessins.

Note sur les Transitions des liens

On va décrire ci-dessous les différentes transitions apparaissant pour des liens Créés, des liens Dérivés, des liens Suggérés et des liens Archive.

a. Transitions des liens Créés (figure 101)

1. Création d'une Référence (nouveau Contenu, Contenu pris du Contenant associé ou d'un autre Contenant)

2. L'Utilisateur veut la supprimer, celle-ci est d'abord mise en Archive (effet : lien Archive transition 1)

3. Remettre dans le Profil une Référence Retirée (« restore ») (effet : lien Archive transition 2)

4. Au cas où le Compteur de Dérivation tombe à zéro, la Référence peut être Supprimée effectivement (Supprimée aussi dans l'Archive – Transition 2) ainsi que son lien.

b. Transitions des liens Dérivés (figure 102)

1. Soit une Référence est (« manuellement ») dérivée d'un autre Profil, soit une Référence suggérée est entérinée. En même temps, le lien Suggéré correspondant (le cas échéant) passe à Consulté (Transition 5).

2. L'Utilisateur peut ensuite la Retirer. Auquel cas elle est d'abord mise en Archive (lien Archive Transition 1). Le lien Suggéré correspondant (le cas échéant) reste à Consulté.

3. « Restauration » (retour de l'Archive). Transition 2 pour le lien Archive.

4. Au cas où le Compteur de Dérivation tombe à zéro, la Référence est supprimée effectivement et son lien l'est donc aussi (supprimée aussi dans l'Archive : Transition 2). En même temps, le lien Suggéré correspondant (le cas échéant) est supprimé (Transition 4).

c. Transitions des liens Suggérés (figure 103)

On notera tout d'abord que le lien Suggéré Dérivé en est un cas particulier.

1. Un Profil reçoit une Référence (un Contenu est suggéré). Le lien Suggéré dans l'état Suggéré n'est pas stocké de manière permanente, sauf dans le cas du Référencement (puisque dans ce cas la suggestion ne découle pas de la dérivation du Profil et n'est donc pas une information redondante).

2. L'Utilisateur le refuse ou l'ajoute (s'il l'ajoute : transition 1 du lien Dérivé). Le lien Suggéré est alors stocké de manière permanente (dans l'état Consulté). Le système pourra ainsi éviter de suggérer à nouveau cette Référence à la prochaine connexion

3. « Restauration ». Le lien stocké de manière permanente est simplement supprimé dans la BD des Profils sauf dans le cas du Référencement. L'élément pourra ainsi être suggéré à nouveau.

4. La source étant effectivement supprimée (lien Archive - 2), son lien est supprimé. S'il existe, le lien Dérivé correspondant effectue la Transition 4.

d. Transitions des liens Archive (figure 104)

1. Référence mise en Archive (cause : transition 2 des liens Créés et Dérivés).
2. Lien Archive supprimé, soit parce que la Référence est restaurée (transition 3 des liens Créés et Dérivés), soit parce que le Compteur de Dérivation tombe à zéro (plus aucune autre Référence ne pointe sur la Référence attachée à ce lien), dans ce cas : Transition 4 pour les lien Suggéré et Dérivé ou pour le lien Créé.

iv) Archivage et Restauration

Partons de la situation illustrée sur la figure 105, dans laquelle un Profil A possède une Référence R' sur une Référence R d'un Profil B.

Quand la référence R' est Retirée elle va en Archive, comme illustré sur la figure 106.

Mais dès que plus aucune Référence ne pointe sur elle (dès que le Compteur de Dérivation de R' passe à zéro), R' peut être supprimée effectivement. Ceci est illustré sur la figure 107.

Ce premier perfectionnement de la présente invention présente, outre les avantages de la forme de réalisation de base, toute une série d'avantages supplémentaires, et principalement le fait de pouvoir dériver des Profils, et donc de bénéficier de « canaux de communication » automatiques entre Utilisateurs. Plus précisément :

- alors que dans la forme de réalisation de base, l'Utilisateur pouvait consulter les Contenants de la BD Commune et s'abonner à (être automatiquement informé de) leurs nouveaux Contenus, le premier perfectionnement permet à l'Utilisateur de consulter et de s'abonner aux Profils qu'il a dérivé. Il exploite ainsi l'expertise (de regroupement sélectif et de catégorisation d'informations) matérialisée dans ces Profils ; le schéma de la figure 92 illustre qu'un Contenu, ajouté par un Utilisateur UB, est communiqué à un Utilisateur UA automatiquement via leurs Profils

respectifs, grâce au fait que le Profil de UA (Profil A) est dérivé de celui de UB (Profil B) ; autrement dit, les Références ajoutées dans un Profil d'un Utilisateur UB, lequel Profil est dérivé par un Utilisateur UA, sont par défaut automatiquement communiquées à UA.

- les Références ajoutées dans un Profil Dérivé sont automatiquement suggérées au(x) propriétaire(s) du (des) Profil(s) d'origine. Au cas où ce(s) dernier(s) les valide(nt), son (leurs) propre(s) Profil(s) se trouve(nt) enrichi(s) de ces Contenus, et, en vertu de ce qui précède, ces derniers sont alors automatiquement propagés aux autres Utilisateurs ayant dérivé le(s) même(s) Profil(s) et qui s'y sont abonnés. ; le schéma de la figure 95 illustre la communication de UB à UC via le Profil de UA, Les Profils de UB et UC ayant été dérivés du Profil de UA.

- enfin, grâce à la fonctionnalité d'Archive, un Contenu dérivé dans un Profil reste vivant même quand le Contenu d'origine est supprimé.

D. Deuxième perfectionnement (Niveau 3)

Selon ce deuxième perfectionnement, les Utilisateurs ne partagent plus une structure commune de Contenants. Autrement dit, chaque Utilisateur peut avoir une structure de Contenants différente de celles des autres Utilisateurs.

Ces structures de Contenants sont mémorisées dans des « Bases de Données (BD) Personnelles » des Utilisateurs et non dans la BD Commune, qui d'ailleurs n'existe plus en tant que telle mais en tant qu'ensemble de fichiers (« Contenants »).

Dans le même esprit que ce qui précède, une BD Personnelle est composée de Références plutôt que des Contenants et Contenus eux-mêmes. En effet, pour chaque Contenant, tous les Contenus (sauf ceux qui sont confidentiels) sont mémorisés en dehors de la BD Personnelle, dans un fichier commun (« Contenant ») . Dans la BD Personnelle, les références à ces Contenus forment un « Profil ». Certains de ces

Profils peuvent être publiés à l'ensemble ou à un sous-ensemble des autres Utilisateurs (Profils Non Confidentiels - voir plus haut).

Les Profils contiennent des Sous-Profils et forment ainsi une structure arborescente illustrée sur la figure 108 (ou encore une structure de graphe), qui peut être différente d'une BD Personnelle à une autre.

De manière primitive, cette structure peut être représentée par un diagramme de classe UML , comme illustré sur la figure 109.

Ici encore, le système offre une garantie d'anonymat : Les Utilisateurs, pour un Contenant donné (accessible à travers un Profil de leur BD Personnelle), peuvent consulter (ou être abonné à) tous les Contenus ajoutés par les autres Utilisateurs dans ce Contenant, sans pouvoir déterminer quel Contenu figure dans quelle BD Personnelle.

Ici, deux Profils associés à un même Contenant peuvent avoir une dénomination différente. On appelle « Catégorie » le nom d'un Profil. Comme la Catégorie d'un Contenant dépend du choix de l'Utilisateur, la classe Catégorie est associée à la classe « Profil » (ou en constitue un attribut) et non pas à la classe « Contenant ».

On va maintenant décrire la façon dont l'arborescence est construite par chaque Utilisateur , c'est-à-dire la façon dont les différents Sous-Contenants sont créés dans les Profils, ainsi que l'incitation qui existe pour un Utilisateur à dériver un Profil.

a) Nouveau Contenant

L'Utilisateur peut librement créer un nouveau Contenant, en tant que sous-profil dans un Profil de sa BD Perso. S'il est Non Confidentiel, celui-ci donnera lieu à un nouveau fichier (Contenant-1) dans la Partie Commune, comme illustré sur la figure 110.

b) Contenant déjà existant

Alternativement, l'Utilisateur peut dériver un Profil Non Confidentiel d'un autre Utilisateur ou un Profil de sa propre BD Personnelle. Ce Profil est ajouté en tant que Sous-Profil dans un Profil de sa BD Personnelle qu'il a choisi (il a par exemple inséré le Profil dérivé dans ce Profil par la technique du « glisser-déposer »). Les éléments (Contenus et Sous-Profils) que le Profil source contient sont automatiquement suggérés dans ce nouveau Sous-Profil.

On notera ici que le fait de dériver un Profil n'engendre pas automatiquement la dérivation de toute la structure sous-jacente. Autrement dit, les Sous-Profils ne sont pas dérivés d'office, mais seulement si l'Utilisateur les ajoute explicitement. En effet, dans le cas contraire, le processus de dérivation pourrait rentrer dans une boucle infinie.

c) Incitation l'Utilisateur à dériver un Profil

i) Lien Profil-Profil Créé

L'Utilisateur peut décider de dériver un Profil suite à sa consultation, par sa propre initiative, des Profils (non confidentiels) mis à sa disposition par les autres Utilisateurs.

ii) Lien Profil-Profil Suggéré ou Dérivé

Grâce au processus de suggestion descendante indiqué plus haut à propos du premier perfectionnement, les nouveaux Sous-Profils ajoutés dans un Profil P1, duquel un Profil P2 a été dérivé, se retrouvent automatiquement dans P2, en mode suggéré. Inversement, grâce au processus de suggestion remontante, un nouveau Sous-Profil

ajouté dans P2 est suggéré dans P1. Quand un nouveau Sous-Profil apparaît ainsi, l'Utilisateur peut l'ignorer (c'est le cas par défaut), le garder (l'ajouter) ou le refuser.

iii) Lien Profil-Profil Référence (« Suggestion Manuelle »)

Ce processus est appelé « Référencement ». Un Utilisateur UA peut, de sa propre initiative, suggérer une Référence R1 ou un Profil P1 à un Utilisateur UB , pour qu'il soit dérivé en tant que Référence R1' ou profil P1' sous un Profil P2 (Non Confidentiel) de la BD Personnelle de UB.

Ces différents liens apparaissent sur le diagramme de classe de la figure 111.

Outre les avantages mentionnés plus haut à propos de la forme de réalisation de base et du premier perfectionnement (niveaux 1 et 2), ce second perfectionnement de l'invention ajoute d'autres avantages :

- tout d'abord, différents Utilisateurs peuvent avoir une structure différente de Contenants ; l'Utilisateur est ainsi maître de la catégorisation de ses données et n'est pas contraint par une structure commune figée ; dans ce nouveau cadre, le système :

  * exploite toujours le fait que les Contenants sont partagés (avantage du niveau 1), et

  * permet des dérivations non seulement de Contenus mais aussi de structures arborescentes de Contenants.

- ensuite, la dérivation de Profil ou de Contenu peut se faire, non seulement par l'initiative de l'Utilisateur qui dérive, mais aussi par l'initiative d'un autre Utilisateur qui souhaite que ses Profils ou Contenus soient dérivés dans un Profil externe (mécanisme de Référencement) et qui pour cela les suggèrent « manuellement » (par opposition aux suggestions automatiques décrites plus haut.

Ce dernier avantage est considérable ; grâce au Référencement, le système peut s'étendre et voir sa Communauté d'Utilisateurs augmenter toujours plus, sans véritable risque de « pollution » à cause de l'introduction d'informations de mauvaise qualité. En effet, bien que de telles informations puissent effectivement être introduites dans le système, le nombre d'Utilisateurs qui y accéderont ne sera pas significatif. Car pour que l'on y accède à grande échelle, il faudrait qu'elles soient Référencées dans les Profils publics « à grande audience », c'est-à-dire connus par un grand nombre d'Utilisateurs et largement consultés et/ou dérivés. Or, les propriétaires de ces Profils publics ne sont pas obligés d'accepter un Référencement suggéré par n'importe qui et peuvent le filtrer. Ils filtreront ainsi les mauvaises informations sous peine de dévaloriser leurs propres image et la confiance de leur « audience ». Les Utilisateurs du système jouent donc eux-mêmes le rôle de « modérateurs » décentralisés. Le système est ainsi auto-régulé.

E. Troisième perfectionnement (niveau 4)

Ce troisième perfectionnement consiste en une évolution du deuxième pour prendre en compte l'assemblage, dans un Profil d'un Utilisateur, de plusieurs ensembles de Contenus pour un même Contenant.

Selon ce perfectionnement, les Profils du Niveau 3 sont remplacés par des « Classeurs ». Un Classeur est associé à un Contenant et contient (zéro, une ou) plusieurs « Pages ». Chaque Page rassemble un ensemble de Contenus.

Un Classeur contient donc des Pages et des sous-classeurs.

La structure arborescente est ainsi représentée sous la forme d'une hiérarchie de « Classeurs », comme illustré sur la figure 112.

Les classeurs, les pages et les Références peuvent être dérivés. Les liens Profil-Contenu sont remplacés par des liens Classeur-Page et Page-Contenu. L'Archive de

Profil est remplacée par une Archive globale pour chaque Base de Données Personnelle.

Cette structure est spécifiée par le diagramme de classe illustré sur la figure 113.

Les Classeurs et les Pages (Non Confidentiels) d'une BD Personnelle peuvent être dérivés d'une autre BD Personnelle, voire d'un autre endroit de la même BD Personnelle.

Les avantages des niveaux 1 à 3 subsistent ici.

On va maintenant décrire une application pratique de ce troisième perfectionnement.

Un Utilisateur possède un classeur de base qui lui est « propre », dans lequel il peut réaliser différentes actions, et en particulier :

a) Création de sous-éléments « propres à l'Utilisateur »

L'Utilisateur constitue son Classeur de ses propres Sous-Classeurs, Pages et Calques (on décrira ces deux derniers objets dans la suite) grâce à un éditeur de documents intégré au système.

b) Création de sous-éléments « dérivés »

L'Utilisateur insère dans son Classeur des éléments récupérés dans d'autres Classeurs.

c) Suppression de sous-éléments « propres à l'Utilisateur »

L'Utilisateur supprime ses propres éléments.

d) Acceptation ou Refus de sous-éléments « dérivés »

L'Utilisateur rend visible ou non les sous-éléments « implicites » des éléments « dérivés », c'est à dire les sous-éléments des éléments récupérés dans d'autres Classeurs.

Ces informations sont stockées dans une base de données par exemple de type SQL, telle qu'illustrée sur la figure 114.

On notera que les champs de référencement REFelement, REFstyle, REFcontenu sont ici des adresses de type « url ». Les pages vers lesquelles on pointe renvoient ainsi le code désiré, et en particulier les données de Contenu en format HTML, les données de Classeur en format XML, etc.

Comme on le voit sur la figure 114, les Classeurs, Pages et Calques sont des « éléments » reliés hiérarchiquement par des tables « Lien ». leurs définitions ne se distinguent que par un attribut en moyenne (IDcontenant par un classeur, REFstyle pour une page, IDcontenu pour un calque).

On va décrire ci-dessous, dans ce contexte, des exemples d'actions.

a) Prologue : création d'Utilisateur

Avant toute action de l'Utilisateur, il faut bien que celui-ci soit défini. Son existence est inscrite dans une table « Utilisateurs », telle qu'illustrée sur la figure 115.

(Remarque : à la création d'un nouvel Utilisateur, on crée également un « Classeur de Base » et un élément pour « Archive » (élément de type particulier pour stocker tous les éléments en voie de suppression, comme décrit plus haut).

b) Action 1 : création d'élément/sous- élément « propre à l'Utilisateur »

La figure 116 illustre une structure hiérarchique de Classeurs et une Table associée, pour un Utilisateur donné (ici l'Utilisateur No. 1 appelé « Demo »).

Les différents « éléments » de la Table de la figure 116 sont « propres » à cet Utilisateur, en ce sens qu'ils ne font pas référence à un autre élément (le champ « REFelement » est toujours égal à la valeur « Null », c'est-à-dire vide).

On distingue les 3 types d'éléments suivants, reconnus par la valeur de la variable Type de la Table de la figure 116.

i) Les Classeurs (type 1)

Dans le présent exemple, les éléments Nos. 1 et 2 (c'est-à-dire dont les valeurs de IDelement sont respectivement égales à 1 et 2) sont des Classeurs appelés « Demo 0 » et « Demo 0 1 », le Classeur No. 1 étant d'ailleurs le Classeur de Base de l'Utilisateur en question.

ii) Les Pages (type 2)

Dans le présent exemple, les éléments Nos. 3 et 7 sont des Pages de l'Utilisateur.

iii) Les Calques (type 3)

Dans ce même exemple, les éléments Nos. 4 et 5 sont des Calques de l'Utilisateur.

Les liens de parentés sont inscrits dans une Table « Liens », telle qu'illustrée sur la figure 117. Cette Table montre que l'élément No. 1 (« Demo 0 ») est le Parent des éléments Nos. 2 et 6 (« Demo 0 1 » et « Demo 0 2 »)

c) Action 2 : création d'élément/sous-élément « Dérivé »

La figure 118 montre un exemple d'une structure arborescente de classeurs incluant un Classeur « Dérivé » (nommé ici « Aline favoris »), et de la Table associée.

Un élément dérivé résulte de la récupération d'un élément par un autre Utilisateur que son concepteur original. C'est un élément « propre au récupérateur » auquel est ajouté une référence à l'élément original.

L'exemple de la figure 118 montre que l'élément No. 9 (« Aline favoris ») est dérivé de l'élément No. 2 (« Demo 0 1 »), car la valeur de REFelement est ici égale à 2.

Le Classeur No. 9 possède alors implicitement les sous-éléments du classeur No. 2 . Ces sous-éléments devront être Acceptés ou Refusés (voir plus loin action 4) pour que les Utilisateurs qui dériveront le classeur No. 9 les voient ou ne les voient pas, selon le cas.

Comme dans un élément « propre à l'Utilisateur », l'Utilisateur de l'élément dérivé peut ajouter ou retrancher d'autres éléments selon son choix ; ses ajouts et suppressions seront d'ailleurs suggérés par la suite à l'auteur de l'élément original.

(Remarque : Lorsqu'un Utilisateur crée un élément « dérivé » (comme ici le Classeur « Aline favoris »), le compteur de dérivation de l'élément original (variable Compteur » est automatiquement incrémenté (comme ici pour le classeur « demo 0 1 »). Ceci permet de savoir si la suppression de l'original est possible ou si cet original doit être mis en archive lorsque son auteur le masque.

d) Action 3 : Suppression d'éléments « propres à l'Utilisateur »

i) Suppression d'un sous-élément « propre » non dérivé

Considérons l'exemple où l'Utilisateur « demo » désire supprimer le sous-élément « Demo 0 2 ». D'après le Compteur de Dérivation, celui-ci n'est utilisé comme référence par personne. On peut donc supprimer le lien entre « Demo 0 » et « Demo 0 2 », mais aussi supprimer « Demo 0 » et tous ses sous-éléments dont la valeur du Compteur de Dérivation est nul.

Le résultat de cette suppression est illustré sur la figure 119.

ii) Suppression d'un sous-élément « propre » dérivé au moins une fois

Considérons ici l'exemple où l'Utilisateur « demo » désire supprimer le sous-élément « Demo 0 2 ».

Or, d'après la valeur du Compteur de Dérivation, celui-ci est utilisé comme référence par un autre classeur (en l'occurrence le classeur « Aline favoris »). On peut donc supprimer le lien entre « Demo 0 » et « Demo 0 1 », mais on ne peut pas supprimer l'élément « Demo 0 1 » lui-même.

C'est pourquoi, celui-ci est alors apparenté au classeur « Archive » de l'Utilisateur (pour ne pas engendrer un orphelin). Il y restera jusqu'à ce que la valeur du Compteur de Dérivation tombe à zéro.

La situation est illustrée sur les figures 120a à 120c. Ainsi la figure 120c montre que l'élément n°2 (Idenfant = 2) n'appartient plus vraiment à l'élément No. 1 (Idparent = 1), car son attribut « Operation » a la valeur zéro.

Ce même élément est en revanche ajouté à l'élément « archive » n° 12 (Idparent = 12) de l'Utilisateur, dont il devient un sous-élément.

Le fait que ce sous-élément soit rattaché à un élément « archive » permettra de pouvoir encore le visualiser le cas échéant le restaurer à l'initiative de l'Utilisateur, comme on va le voir ci-dessous.

iii) Restauration d'un sous-élément « archive »

Pour restaurer un sous-élément « archive », l'Utilisateur consulte l'élément « archive » et désigne le sous-élément à restaurer. Il suffit alors de remplacer le « 0 » par un « + » dans le « lien » correspondant au sous-élément. Puis on supprime le lien entre cet élément et l'élément « archive ». Ceci est illustré sur la figure 121.

On observera que l'on pourrait mettre également en archive les éléments supprimés dont le compteur de dérivation est nul pour permettre leur restauration... Dans ce cas, la « vidange » se ferait de la même manière qu'avec la « corbeille » d'un explorateur de fichiers. Les éléments ne pourront cependant être « vidangés » tant que leur compteur de dérivation est non nul.

e) Action 4 : Acceptation/Refus de sous-éléments « implicites »

Les éléments dérivés possèdent implicitement les sous-éléments de l'élément original.

Ainsi, à chaque ouverture de son élément dérivé, l'Utilisateur verra apparaître et disparaître des sous-éléments au gré de l'Utilisateur de l'élément original. Ainsi la figure 122 illustre le cas où un classeur n° 9 « Aline favoris » (Idelement = 9) est dérivé d'un classeur n° 2 (REFelement = 2), et possède donc implicitement les sous-éléments de ce dernier, tels que les éléments illustrés sur la figure 123, dont la variable Idparent est égale à 2.

On va décrire ci-dessous comment ces sous-éléments peuvent être Acceptés ou Refusés.

i) Acceptation d'un sous-élément « implicite »

L'Utilisateur doit accepter un sous-élément « implicite » (sous-élément de l'original) pour ne pas risquer de le voir disparaître et pour le proposer aux Utilisateurs qui dériveront ou qui ont dérivé à leur tour l'élément dérivé (par exemple, si dans l'exemple ci-dessus un troisième Utilisateur crée un classeur dérivé du classeur n°9).

L'acceptation d'un sous-élément « implicite » est en fait une dérivation de ce sous-élément dans le classeur dérivé. En base de données, cela revient donc à créer un classeur dérivé et à en faire un enfant de l'élément dérivé.

Exemple d'acceptation du sous-classeur implicite « Demo 0 1 1 »

La figure 124 illustre la création de l'élément n°15 dérivé du n°13, à savoir « Demo 0 1 1 », et la figure 125 illustre la création du lien de parenté entre les éléments. L'Utilisateur exprime ainsi son intérêt à l'auteur original pour cet élément. De plus, l'auteur original pourra ainsi lui suggérer par la suite des ajouts et retranchement de sous-éléments.

ii) Refus d'un sous-élément « implicite »

L'Utilisateur doit refuser un sous-élément « implicite » (sous-élément de l'original) lorsqu'il ne souhaite pas le voir réapparaître à chaque ouverture de son classeur.

Le refus d'un sous-élément « implicite » est en fait un lien de parenté spécial entre ce sous-élément et le classeur dérivé qui permet de signifier le refus. En base de données, cela revient donc à créer un lien entre l'élément dérivé et le sous-élément avec la valeur « - » dans le champ Operation.

Exemple de refus du sous-classeur implicite « Demo 0 1 2 »

Cet exemple particulier est illustré sur les figures 126 et 127, la figure 127 montrant la valeur « - » du champ Operation.

L'Utilisateur exprime ainsi son désintérêt à l'auteur original pour cet élément. En outre, lorsque l'élément original est réellement supprimé, ce lien n'aura plus d'utilité et pourra donc être également supprimé.

On notera ici que la gestion des classeurs et des éléments qu'ils contiennent peut se faire avantageusement selon une interface utilisateur telle qu'illustrée sur la figure 127a. Cette interface utilisateur comprend à gauche deux fenêtres ou cadres superposés Cadre1 et Cadre2, le cadre supérieur Cadre1 affichant une structure arborescente (par exemple du type « Explorateur Windows » - marque déposée de Microsoft Corp.) du classeur de l'utilisateur, tandis que le cadre inférieur Cadre2 contient, avec le même type de représentation, le classeur d'un tiers en train d'être consulté à distance par ce même utilisateur. On peut ainsi facilement naviguer d'un Contenant à l'autre, les Contenus du cadre actif étant affichés dans un cadre plus grand Cadre3 prévu dans la partie droite de l'écran. Ces contenus (pages) sont affichés avec le style imposé par le cadre couramment choisi. Des onglets permettent également de naviguer dans la structure.

Cette interface permet notamment :

- d'importer à partir du classeur tiers, par la technique du « glisser-déposer » du cadre Cadre2 vers le cadre Cadre1, tout ou partie des Contenants (inclus les Contenus associés) ou des contenus individuels, sachant que l'importation d'un ou de plusieurs Contenants peut impliquer une procédure d'acceptation sélective, de façon collective ou individuelle, des Contenus associés ;

- de proposer au tiers, toujours par la technique du glisser-déposer, mais cette fois-ci de Cadre1 vers Cadre2, tout ou partie de son propre classeur, pour mettre en œuvre la fonction décrite plus haut de suggestion manuelle ou « Référencement » ;

- pour ce qui concerne le cadre d'affichage Cadre3, une série de boutons et/ou d'onglets est avantageusement prévue pour faire défiler les pages, changer de calque, etc..

Enfin on notera ici que les opérations d'acceptations/refus de suggestions peuvent s'effectuer dans des boîtes de dialogue spécifiques, soit lorsque l'utilisateur accède à un classeur contenant des éléments nouvellement suggérés depuis sa dernière connexion, soit lorsque l'utilisateur accède au Contenants destinés à accueillir les éléments suggérés.

Par ailleurs, ces aspects de l'invention permettent d'exploiter une nouvelle technologie de "multi-portail" sur l'Internet, qui permet :

- à des fournisseurs d'informations, tels que des magazines grand public par exemple, de créer et mettre à jour leurs informations directement sur l'Internet,

- l'accès aux informations par de multiples portes d'entrées, à partir desquelles les informations peuvent être présentées de manière différente ; par exemple, chaque magazine peut offrir un accès au multi-portail via son propre nom de domaine de type « www.telmagazine.fr »,

- aux fournisseurs d'information de « fidéliser » l'internaute en jouant sur sa fibre de collectionneur ; l'internaute peut en effet constituer des collections d'information dans un classeur personnel en ligne,

- aux fournisseurs de produits, d'offrir des éléments de contenu décrivant leurs produits (biens ou services) : en quelques clics de souris, ils les référencent dans les

magazines en ligne, communiquent ainsi leurs nouveautés et promotions et en récoltent des transactions commerciales,

- aux magazines de se transformer en mini-annuaires (comme les portails ou annuaires connus aujourd'hui sur l'Internet) et de percevoir une commission sur les visites et ventes engendrées,

- enfin, aux internautes de consulter les magazines et les offres de produits : ils peuvent glisser dans leur classeur personnel des éléments de contenu qu'ils retrouvent à chaque connexion, ces éléments de contenu étant mis à jour et complétés ; le fait de glisser une information dans le classeur personnel représente implicitement une " déclaration d'intérêt " qui leur permet de se faire suggérer automatiquement des informations apparentées.

Ainsi un nouveau magazine en ligne rejoignant un multi-portail existant pourra exploiter le trafic qui y est déjà présent.

Section 2 – Auto-épuration du Système

On va maintenant décrire en détail une fonction de la présente invention permettant de prendre en compte la qualité des suggestions pour aboutir, par une notion de « degré de confiance », à limiter les effets de suggestions indésirables. Cette section est à rapprocher de la section « Coefficients de Probabilité d'Acceptation » du chapitre II, sachant que l'acceptation au niveau des répertoires décrite dans cette section du chapitre II est plus large que l'acceptation au niveau des utilisateurs telle que décrite dans la présente section.

Introduction

Le système met en œuvre un double principe d'épuration.

Premièrement, comme mentionné à propos de la suggestion « manuelle » dans la section 1 de ce même chapitre, la diffusion des informations entre Utilisateurs peut s'effectuer par « Référencement » d'un Profil d'un Utilisateur donné dans les Profils (Classeurs ou Pages) d'autres Utilisateurs dont il espère capter l'audience (notamment pour qu'à partir de Profils Non Confidentiels d'autres utilisateurs, le public puisse trouver les informations propres audit Utilisateur donné). Le système permet ceci de manière décentralisée. Le Référencement est une suggestion et peut donc être filtré par le destinataire, qui joue ainsi le rôle d'épurateur. Ainsi, ceux qui fournissent des informations de mauvaise qualité perdent leur crédibilité, et ont tendance à ne plus faire l'objet de dérivations vers d'autres utilisateurs. Leur audience baisse donc.

Deuxièmement, le maintien du crédit qu'accorde chaque Utilisateur peut-être assisté par le système de manière automatique. Par apprentissage automatique et transparent, basé sur l'expérience, le système construit et maintient une notation de contribution révélatrice d'un "degré de confiance" que peut accorder un Utilisateur aux informations fournies par un autre Utilisateur.

Ce degré de confiance peut être déterminé de manière indirecte en suivant une "chaîne" d'Utilisateurs dans laquelle chaque Utilisateur a pu déterminer le degré de confiance pour le suivant (c'est à dire par transitivité).

Le degré de confiance ainsi obtenu permet de filtrer automatiquement les suggestions de contenu ayant une forte probabilité de ne pas être pertinentes.

Il est important de noter ici que l'identité (ou plutôt le pseudonyme) de l'Utilisateur qui est à l'origine de la suggestion est connu du système mais pas des autres Utilisateurs.

Détails du procédé d'auto-épuration

Pour chaque type d'action (accepter, refuser, déplacer, etc.), on prévoit selon ce perfectionnement de l'invention le recours à une notation de contribution, c'est-à-dire à un système d'attribution de points positifs ou négatifs, sur la base du fait que l'Utilisateur B qui reçoit des points positifs par l'Utilisateur A augmente d'autant la confiance que lui accorde ce dernier. De même, il diminue sa confiance quand il reçoit des points négatifs.

L'Utilisateur A peut ainsi automatiquement filtrer (dans ses requêtes) les suggestions faites par des Utilisateurs dont le degré de confiance est inférieur à un certain seuil.

L'intérêt de cette fonction réside surtout dans le fait que l'évaluation (ou notation) se fait dans le cadre l'utilisation courante de l'outil, de manière transparente.

Par exemple, suite à une requête, un Utilisateur A reçoit un Contenu suggéré par un Utilisateur C qu'il n'a pas encore eu l'occasion d'évaluer. Dans le cas où l'Utilisateur C a un degré de confiance négatif chez l'Utilisateur B, et que ce dernier a un degré de confiance positif chez l'Utilisateur A, alors, par transitivité, l'Utilisateur C obtient des points négatifs chez l'Utilisateur A. Ce dernier peut alors automatiquement filtrer les Contenus suggérés par C.

En effet, l'Utilisateur A a confiance dans les suggestions faites par l'utilisateur B. Donc si l'Utilisateur B estime que les suggestions de l'Utilisateur C ne correspondent pas à ses attentes, alors l'Utilisateur A peut être confiant qu'il aurait la même opinion négative.

Concrètement, cette approche peut s'appuyer sur une structure de données incluant une représentation d'un graphe de transitivité entre les différents Utilisateurs (tout en conservant leur anonymat les uns par rapport aux autres).

Avantageusement, cette approche peut s'accompagner de coefficients de pondération en fonction de la longueur de la chaîne de transitivité pour une évaluation donnée, ainsi que d'autres critères et seuils divers.

Par exemple, pour que l'Utilisateur A puisse se fier au jugement de l'Utilisateur B au sujet de l'Utilisateur C, le degré de confiance qu'accorde l'Utilisateur A à l'Utilisateur B doit être supérieur à un certain seuil depuis un certain temps (critère de stabilité).

Ainsi le système construit automatiquement ce qu'on peut qualifier de « réseau de confiance », auto-alimenté et auto-régulé, qui permet d'épurer sélectivement les suggestions inadéquates.

On notera ici qu'il subsiste toutefois un risque : ainsi un Utilisateur malveillant peut s'inscrire dans le système fréquemment, avec des pseudonymes chaque fois différents, dans le but de suggérer des Contenus de mauvaise qualité.

Cet Utilisateur malveillant n'étant évalué par personne au début, ses suggestions ne pourront être filtrés pendant un certain temps.

Pour supprimer ce risque, on pourra filtrer les Utilisateurs récents dont le volume d'éléments suggérés n'a pas atteint un certain seuil, que l'on appelle « seuil de contribution ». Ce filtrage pourra aussi servir à inciter les Utilisateurs à contribuer suffisamment pour que les Contenus qu'ils suggèrent ne soient pas filtrés. Ceci incite à faire vivre le système. Le volume de contribution de chacun pourra être déterminé selon une méthode qui permette d'éviter les contributions reproduites artificiellement pour atteindre le seuil de contribution.

Alternativement, le fait de s'inscrire dans le système d'échange par suggestions peut être payant, ou nominatif via un tiers de confiance qui découragerait de suggérer des Contenus sous des pseudonymes différents.

Section 3 – Techniques de suggestions automatiques

Introduction

Les Profils, tels que décrits dans la section 1, caractérisent les Utilisateurs et peuvent ainsi être exploités en tant que profils d'intérêts (ou de goûts) que l'on confronte les uns aux autres pour en extraire des informations pertinentes et les suggérer automatiquement aux Utilisateurs.

La confrontation d'un profil aux autres peut se faire individuellement (par exemple par les techniques connues de « Filtrage collaboratif », de « Recommandation sur la base de Contenus »), ou sur l'ensemble synthétisé des autres profils de manière compacte (par exemple par les techniques de Réseaux Bayesiens, etc.).

Un apport important de cet aspect de l'invention est de tirer parti de l'ordre des Contenus dans les Profils pour améliorer les suggestions. Si l'on se place au Niveau 4 de la section 1 du présent chapitre, l'utilisateur peut ordonner les Contenus (qui sont affichés dans un certain ordre dans les pages), et les pages entre elles (l'ordre des pages est affiché sous la forme d'un classeur), en fonction de différents critères d'ordre choisis par lui-même (par exemple, critère de préférence globale, critère temporel, critère économique, critère esthétique, etc.) ou une conjonction ordonnée de tels critères.

Dans la mesure où ces critères sont partagés avec d'autres Utilisateurs, le système en déduit des profils ordonnés, améliore ainsi la pertinence des éléments recommandés, insère les éléments recommandés à leur bonne position, et peut même ordonner automatiquement des informations posées en vrac en se basant sur l'ordre adopté par les profils proches. Ces procédés sont décrits dans la section suivante.

Détails des techniques de Suggestion Automatique

a. Détection de profils voisins

Les Contenus sont composés d'éléments (par exemple au format XML) qui ont des attributs et des éléments imbriqués. Afin de détecter, entre deux profils donnés, les Contenus qui peuvent être considérés comme « pratiquement identiques », ils peuvent être analysés selon des règles qui se déclenchent à partir d'un mécanisme de reconnaissance de motifs (« pattern matching »).

Un Contenant peut ainsi donner des poids différents aux attributs des éléments de ses Contenus. Un Contenu sera alors considéré comme « pratiquement identique » à un autre si un ensemble d'attributs suffisamment représentatif (la représentativité sera calculée en fonction de leurs poids respectifs) ont les mêmes valeurs.

Ainsi, il n'est pas nécessaire que deux Utilisateurs aient dans leurs profils des Contenus exactement les mêmes pour qu'ils soient jugés proches.

La proximité entre deux profils Px et Py - ordonnés selon un critère donné - est mesuré en évaluant la quantité de calques que les Utilisateurs A et B ont en commun, avec une pondération par la position de ces calques dans les profils des Utilisateurs A et B respectivement, comme illustré sur la figure 128.

Pratiquement la mesure de proximité va se faire Contenu par Contenu tant que le poids marginal reste supérieur à un certain seuil. Ainsi il suffira comparer un nombre fixé d'éléments dans l'ordre correspondant au critère donné.

L'Utilisateur peut aussi s'appuyer sur un certain nombre d'autres indicateurs tels que par exemple :

- le pourcentage d'informations en commun, comme illustré sur la figure 129 ;
- le nombre de fois qu'un même Contenu est suggéré (à partir de Profils différents).

b. Insertion des Contenus suggérés qui sont acceptés

Lorsqu'un Utilisateur accepte une suggestion de Contenu de la part du système de Recommandation Collaborative, le Contenu est classé au bon endroit selon chaque critère de la configuration du classeur.

On va considérer, en référence à la figure 130, l'exemple suivant : supposons que pour un Contenant donné, des Utilisateurs u1 et u36 ont en correspondance respectivement les pages (« xSheet ») xsA et xsB, dont les Contenus sont ordonnés chacun selon deux critères $\lambda 1$ et $\lambda 2$.

Pour cette catégorie, u1 et u36 sont proches au sens où ils possèdent tous les deux les Contenus c4 et c12. Le système de Recommandation Collaborative fait donc les propositions telles qu'illustrées sur la figure 131, c8 étant suggéré de u36 à u1 et c1 étant suggéré de u1 à u36.

Si ces suggestions sont acceptés par chaque Utilisateurs, les Contenus sont alors insérés en accord avec les positions qu'ils occupent dans leur page d'origine.

Pour le critère $\lambda 1$ de u1, c8 est classé par rapport aux Contenus communs à u1 et u36, il est donc placé avant c4 et c12. Bien sûr, u1 est libre de replacer c8 s'il le juge mal classé en regard du critère utilisé. Un raisonnement symétrique est appliqué pour u36.

Pour le critère $\lambda 2$, les éléments communs à xsA et xsB ne sont pas classés dans le même ordre. Plusieurs conventions sont alors possibles : celle qui semble la plus appropriée est de ne violer aucune relation établie dans le profil de départ. Ainsi, c8 doit être inséré après c12 et c4 dans xsA, et c1 doit être inséré après c12 dans xsB (bien qu'il soit avant c4 dans le profil de départ). Le résultat est illustré sur la figure 132.

Une autre convention pourrait par exemple être d'insérer c1 dans xsA et c8 dans xsB relativement à l'élément commun le plus proche, comme illustré sur la figure 133.

L'insertion d'un Contenu, telle qu'on l'a décrite ci-dessus avec c1 et c8, provoque la mise à jour des profils de u1 et u36 pour les nœuds supérieurs de la taxonomie (profils agrégés). La place où le nouveau Contenu est inséré dans les profils agrégés est fonction de l'endroit où il a été classé lors de son insertion, et du critère utilisé.

Ordonner les Profils automatiquement (Tri de profil par recommandation collaborative)

Comme mentionné précédemment, dans un profil certains des Contenus n'ont pas été explicitement ordonnés. Dans les feuilles, à ceux-ci correspondent des onglets grisés.

Pour chaque profil, les Contenus grisés qui font partie des Contenus en commun avec un profil voisin, mais dans la partition non grisée chez le voisin, sont ordonnés selon l'ordre suggéré par ce voisin.

Section 4 - Graphes de dérivation

Introduction

Rappelons ici que le système décrit dans la section 1 du présent chapitre offre à l'Utilisateur la possibilité de (et une stimulation pour) dériver et réorganiser (ajouter, retirer, modifier, ré-ordonner, déplacer d'un Contenant à un autre, etc.) des éléments d'information non déclarés comme confidentiels. La stimulation vient du fait que l'Utilisateur profite ainsi de l'expertise de toute la chaîne d'Utilisateurs « à l'amont » c'est-à-dire à l'origine de chaque information.

Le système incite donc ses Utilisateurs à l'étendre (à le faire « grossir ») par dérivation, et les Utilisateurs en viennent à partager les mêmes Contenants et Contenus. Le grand avantage offert est ainsi le fait de « parler le même langage »,

tant pour les informations elles-mêmes (Contenus) que pour leurs catégories respectives (Contenants). Or, parler un langage commun (celui-ci étant constitué par l'ensemble des Contenants et Contenus qui se trouvent dans la partie commune du système) facilite grandement la comparaison de Profils. En effet :

- à partir des Contenus, on peut retrouver directement (en accès direct) les Références sur ceux-ci et les Profils qui les contiennent ;
- à partir d'un Contenant associé à un Profil donné, on peut retrouver directement tous les Contenus auxquels ont accédé les Utilisateurs (et par là on peut ainsi retrouver directement les Profils qui les contiennent).

De l'utilisation du système résulte donc un « graphe de dérivation » dynamique, dont le nœuds sont les Profils et les arcs (orientés) sont les dérivations entre Profils effectuées par les Utilisateurs. Il peut s'agir de dérivations de Profils (dérivation d'un Classeur ou d'une Page, si l'on se place au Niveau 4) ou dérivation de Contenus.

L'idée essentielle de cet aspect de la présente invention est de se servir du graphe de dérivation entre Profils pour bénéficier de la localité des comparaisons à effectuer entre Profils.

A ce sujet, l'interface utilisateur du système comprend avantageusement un moyen (bouton, etc.) permettant à un utilisateur consultant un Contenant donné d'un Classeur tiers de consulter directement les contenus désignés dans l'ensemble du contenant de la partie commune qui définit cette catégorie, sans avoir ni à naviguer dans l'arborescence, ni à passer vers un portail Internet ou analogue.

Comment tirer parti du graphe de dérivation entre Profils

En s'appuyant sur le graphe de dérivation (qui évolue dynamiquement en cours d'exécution), chaque Profil (c'est-à-dire chaque nœud du graphe de dérivation) maintient en mémoire une liste des voisins se trouvant dans un certain périmètre (à une distance inférieure à un seuil donné, distance mesurée en nombre d'arcs de dérivation), avec pour chaque voisin, un ensemble d'indicateurs de proximité.

Les indicateurs peuvent être par exemple ceux mentionnés dans la section précédente ou tout simplement des compteurs de Contenus en commun.

On peut en outre réaliser un maintien incrémental des indicateurs de proximité entre Profils : à chaque ajout et retrait de Contenu dans chaque Profil, ces indicateurs sont mis à jour chez tous les voisins dans le graphe. Par exemple, à chaque ajout de Contenu, pour chaque Profil voisin qui le posséderait aussi, le système incrémente l'indicateur « compteur de Contenus en commun » dans les deux Profils en question. A chaque retrait de Contenu, le système décrémente ce même compteur.

Chaque Profil maintient ainsi de façon incrémentale les connaissances qu'il a de la proximité de ses voisins (anonymes bien entendu) et le système peut ainsi effectuer des suggestions automatiquement par Filtrage Collaboratif.

Le seuil de distance peut par ailleurs être ajusté en cours d'exécution par une technique d'apprentissage basée sur un critère de rendement (évaluation des succès des suggestions à différentes distances).

On observera ici que la présente invention permet d'éviter de recourir à des calculs extrêmement lourds de comparaison de profils par leurs contenus eux-mêmes, qui typiquement nécessitent, lorsque le nombre d'utilisateurs est important, des traitements « batch » de durées considérables.

Section 5 – Anonymat et commerce électronique

Architecture

On considère ici une architecture composée de 4 classes d'ordinateurs connectés sur un réseau :

- au moins un poste-client (par exemple de type ordinateur personnel ou terminal de réseau),manipulé par un Utilisateur ;
- au moins un serveur tiers (ST), à savoir serveur de transfert de données (ou dispositif équivalent) ou serveur de confiance ;
- au moins un Serveur de Stockage (SdS) ;
- éventuellement un ou plusieurs Serveurs de Fournisseurs de Produits (SFP).

Les besoins de base sont les suivants :

- Anonymat minimal : l'anonymat des Utilisateurs peut être directement assuré par le ou les SdS qui centralisent les éléments d'information recueillis.

- Anonymat par tiers de confiance : alternativement, l'anonymat peut être réalisé par une architecture incluant un ou plusieurs serveurs supplémentaires qui sont gérés par un ou plusieurs tiers de confiance (ST) et qui sont des "passages obligés" lors des communications en consultation ou ajout d'éléments ; les adresses IP des Utilisateurs, qui consultent les éléments d'informations, sont ainsi cachées.

L'identité (ou le pseudonyme) de l'Utilisateur qui ajoute un élément d'information est transformée (codée, chiffrée) par le serveur tiers, de manière différente à chaque ajout (en plus du fait que son adresse IP est cachée).

La correspondance entre différentes transformations d'une même identité est connue par un ou plusieurs des serveurs tiers, ou par un ou plusieurs serveurs tiers indépendants des premiers (ce sont eux qui notamment peuvent dériver les degrés de confiance).

Les serveurs tiers peuvent servir à fournir un certificat pour témoigner de la fourniture d'un élément d'information par un serveur. Ceci peut être utilisé dans le cadre de divers modèles économiques, comme on le verra plus loin.

Les classeurs pourront aussi être stockés sur un serveur de stockage SdS. Dans ce cas, leurs liens sur les Contenus non-confidentiels (qui forment les profils) seront stockés sur le serveur de stockage SdS après chiffrement sur l'ordinateur personnel (ou un serveur de transfert ST). Ces liens ne pourront pas être déchiffrés par le serveur de stockage, mais seulement sur l'ordinateur personnel ou sur le serveur de transfert.

Les besoins supplémentaires impliqués par le procédé de Recommandation Collaborative sont les suivants :

- Anonymat minimal :

L'anonymat des profils peut être assuré directement par des serveurs de stockage qui jouent aussi le rôle de serveurs de recommandation, c'est à dire qui produisent les profils, les maintiennent et les confrontent.

- Anonymat par tiers de confiance :

Alternativement, des serveurs de transfert peuvent avoir le rôle de produire et maintenir les profils et ensuite de les communiquer à un serveur de stockage après les avoir rendus anonymes, c'est à dire, après leur avoir associé une identité transformée (codée, chiffrée).

Les serveurs de recommandation ne manipulent alors que des profils anonymes et sont incapables de déceler que deux profils appartiennent à un même Utilisateur.

Pour éviter d'associer ensemble différents profils d'un Utilisateur, en reconnaissant une même adresse IP lors de leur transmission (par exemple quand manifestement il n'y a qu'un seul Utilisateur en ligne), on préfèrera utiliser un serveur de transfert différent par profil (autant que possible, c'est à dire dans la mesure où l'on dispose d'un nombre suffisant de serveurs de transfert.

Par ailleurs, l'architecture décrite ci-dessus peut être mise à profit dans le cadre du commerce électronique par Internet, comme on va maintenant le décrire en référence à la figure 134.

Ainsi cette figure montre qu'un serveur de stockage SdS contenant des Profils est capable de faire à un poste utilisateur PC des suggestions d'achat auprès du serveur d'un fournisseur SFP, ceci par l'intermédiaire d'un serveur tiers ST permettant d'assurer l'anonymat des utilisateurs au niveau des Profils. L'achat effectif d'un bien ou d'un service par l'Utilisateur auprès du serveur SFP, qui correspond à une acceptation de la suggestion, peut être prouvé par le serveur tiers ST auprès du serveur SFP par l'intermédiaire du serveur de stockage SdS, tandis que le service rendu par le serveur de stockage SdS et par le serveur tiers ST est dûment rémunéré (commission sur vente) par le serveur du fournisseur SFP (la rémunération $ du serveur tiers ST étant constituée par une fraction de la rémunération $$$ du serveur de stockage.

Section 6 - Abonnements et fidélisation

Selon cet autre aspect de la présente invention, il est intéressant qu'un serveur (par exemple un magazine électronique) propose de nouveaux contenus, sur une base périodique, par exemple bi-hebdomadaire. A l'issue de cette période, les contenus sont archivés comme décrit plus haut, et ne sont donc plus suggérés. De nouveaux contenus sont alors mis en place. On comprend que ceci incite fondamentalement l'utilisateur ou abonné à visiter le serveur au moins une fois dans la période de « validité » des contenus ajoutés (en l'espèce deux fois par semaine), ceci de façon à bénéficier de toutes les suggestions d'ajouts.

Ainsi, les mécanismes de dérivation/suggestion/acceptation présentés dans ce chapitre et dans le chapitres précédents (voir le mode gelé dans le chapitre précédent) permettent d'exploiter la fibre de collectionneur de l'utilisateur. En effet, l'utilisateur qui collectionne des contenus dans un contenant est fortement incité à ne pas

manquer une opportunité d'enrichir sa collection pendant qu'un contenu est encore disponible en ligne. Il veut ainsi retourner visiter le site Internet à la même fréquence que la mise à jour des contenus. L'utilisateur se trouve ainsi fidélisé.

* * *

Chapitre IV - Modèle d'exécution déterministe (Figures 135 à 169)

On va dans ce chapitre introduire un modèle d'exécution qui sera utilisé notamment dans le cadre du partage d'aplications sur l'Internet que nous présenterons dans le chapitre VI.

On dit d'un système qu'il est déterministe si, pour une même séquence d'impulsions passée en entrée, ses objets effectuent toujours les mêmes transitions (au sens de leur diagramme d'état-transition).

Rehercher le déterminisme est le fruit d'une double motivation, permettant d'assurer les propriétés suivantes :

- Pouvoir reproduire de façon identique une même exécution : un utilisateur s'attend à ce que l'application qu'il utilise soit reproductible, c'est-à-dire ait toujours le même comportement.
- Pouvoir reproduire de façon identique une exécution entre plusieurs machines à l'aide d'un mécanisme de réplication par calcul. Ce point sera détaillé dans le chapitre 3 : « Interactions multi-utilisateurs ».

Assurer le déterminisme n'est cependant pas évident pour une application constituée de plusieurs objets « actifs » en même temps. On a alors des objets qui évoluent en parallèle, dans ce qu'on appelle un modèle « multi-threading » selon la terminologie anglo-saxonne. Cette approche, bien que présentant certains avantages, crée du non-déterminisme. En effet, d'une exécution à l'autre, l'occupation du système peut varier, et les tâches effectuées par les objets peuvent prendre plus ou moins de temps.

Une telle approche, où chaque objet évolue selon un déroulement (« thread » selon la terminologie anglosaxonne) indépendant, ne peut donc pas répondre à la contrainte de déterminisme visée. La solution selon la présente invention s'appuie sur l'arbitrage des actions effectuées par les objets. Pour ce faire, les applications se voient dotées d'un objet particulier, constituant un Séquenceur.

La fonctionnalité première du Séquenceur est de réussir à garantir le déterminisme tout en donnant un effet de parallélisme, nous parlerons de quasi-parallélisme. En effet, le Séquenceur permet de donner à chacun des objets à tour de rôle une partie des ressources correspondant à l'envoi d'un message (donc la réalisation d'une action).

Cet objectif est schématisé sur la figure 135 des dessins, où l'on observe que des actions indépendantes de quatre objets A, B, C et D sont organisées dans un ordre déterminé, selon un processus de déroulement unique, allouant tour à tour des ressources à chacun de ces objets.

Section 1 - Cadre

Nous nous référerons à la notion d'*objet* du jargon courant de l'informatique orientée-objet – conformément, par exemple, à l'ouvrage « The Unified Modeling Language (UML) Reference Manual », Rumbaugh, Booch et Jacobson, Addisson Wesley 1999.

Les applications que nous manipulons ici sont composées d'objets qui interagissent. Ces objets, et leurs relations, peuvent être définis à l'aide d'un *diagramme de classe* UML, dont un exemple est donné à la figure 136.

On rappellera ici que la notion de « Multiplicité » (par exemple le chiffre 3 associé à l'objet D indique le nombre d'objets de la classe qui peuvent être créés en cours

d'exécution, et que la notion de « Rôle » (par exemple « theC » dans le lien entre B et D) indique le nom sous lequel un objet connaît un autre objet ; ici l'objet de la classe B connaît l'objet de la classe C sous le nom theC. C'est grâce au nom theC, que l'objet de B peut communiquer avec l'objet de C.

A chaque objet défini (via sa classe) dans un diagramme de classe, on peut associer un comportement au moyen d'un *diagramme d'état-transition*. Un tel diagramme décrit le comportement des objets d'une classe donnée sous la forme d'une machine à état.

Ainsi, à un instant donné, un objet est dans un état précis. Les objets changent d'état selon des transitions qui peuvent (ou non) être provoquées par des événements.

Pour prendre un exemple très simplifié, on peut modéliser un interrupteur (« Switch ») et une lampe (« Lamp »), comme illustré sur la figure 137 des dessins.

Selon le modèle illustré sur la figure 138, un objet Switch, qui se trouve dans son état initial « Idle », peut recevoir un événement « TurnSwitch » qui déclenche une transition de l'état « Idle » vers le même état, lors de laquelle un message « TurnLamp » est envoyée à l'objet « myLamp » (de classe « Lamp »). Ce message « TurnLamp » devient, pour la lampe qui la reçoit, un événement déclenchant une transition de « Off » à « On », ou vice-versa, selon l'état dans lequel elle se trouve.

Le fait de pouvoir envoyer un message « TurnLamp » à un objet de la classe « Lamp » implique que cette classe expose la *méthode* correspondante. L'envoi du message revient en effet à appeler la méthode correspondante. (Noter que les messages peuvent avoir des arguments comme les méthodes). La représentation de la classe « Lamp » pourvue de la méthode TurnLamp est illustrée sur la figure 139.

Notons aussi que, dans la suite de la présente description, nous ne traitons pas les *attributs*, car nous les assimilons à des objets liés par des relations de compositions

(c'est-à-dire des sous-objets), ce qui permet de les inclure dans le cas général du traitement des objets. (Les attributs se distinguent des objets en général, par le fait qu'ils n'ont pas de liens avec d'autres objets ; ceci ne contredit pas la généralisation ici faite). Pour mémoire, la figure 140 illustre l'équivalence entre une classe « Classe1 » dotée d'un attribut « Attribute1 (de classe Classe2) », et le lien entre les deux classes « Classe1 » et « Classe2 ».

Enfin, dans la suite de la présente description, nous ne traitons pas les cas d'actions spécifiées sur une transition, car celles-ci peuvent être transformées en une forme équivalente, dans laquelle les actions sont spécifiées dans un état intermédiaire qui a pour seule transition sortante une transition automatique. Pour la classe *Switch* de l'exemple précédent, une telle équivalence est illustrée sur la figure 141.

Dans la suite de la description, nous considérons les diagrammes d'état-transition de la forme illustrée sur la figure 142. Le point noir désigne un « Pseudo Etat », qui symbolise l'état initial, c'est à dire l'état où se trouve l'objet juste après sa création. Un « Evénement » (ici « évènement1 » pour la transition entre Etat0 et Etat1) déclenche la transition à laquelle il est associé dès que l'objet le reçoit. Si aucun événement n'est associé à une transition, celle-ci a lieu dès que l'objet a terminé d'exécuter ses actions (elle est alors appelée « tansition automatique »). Enfin les « Actions » (ici par exemple « Entry/theC.evt » sont les tâches que l'objet peut effectuer dans un état donné.

Les transitions sont vues comme *instantanées* alors que les états ont généralement une durée. A chaque état E d'un objet correspond ainsi une partie de son comportement. Cette partie est décrite par un ensemble d'actions effectuées par l'objet dans l'état E. Ces actions permettent de déclencher des transitions d'autres objets, par l'émission de l'événement associé. Dans l'exemple ci-dessus, B déclenche la transition associé à l'événement « evt » chez C (identifié par le rôle « theC »).

Au sein d'un état, les actions peuvent être de deux types :

- *On Entry* :

  Ce sont les actions que l'objet effectue *en entrant* dans un état. Cette séquence d'actions ne peut en aucun cas être interrompue, elle a un caractère atomique.

- *Activity* :

  Ce sont les actions effectuées par l'objet une fois qu'il est entré dans son nouvel état (donc une fois la séquence atomique des actions *On Entry* terminée). Cela constitue un ensemble d'actions, chaque action étant atomique. La réception d'un événement associé à une transition peut par contre interrompre cette partie (entre deux actions atomiques).

Un état peut n'avoir aucune action *On Entry* comme il peut en avoir plusieurs, il en est de même pour les actions *Activity*.

Un état sera donc représenté de la façon illustrée à titre d'exemple sur la figure 143. Notons que « Do » spécifie ici les actions de type *Activity*).

On va illustrer les actions *Activity* d'un état par prolongement de l'exemple de l'interrupteur et de la lampe présenté précédemment, illustré à nouveau sur la figure 144.

Dans cet exemple, et comme illustré sur la figure 145, la lampe n'a aucune action de type *On Entry* dans son état On. Dans le cadre de la partie *Activity* (messages « Do/ ») de cet état, la lampe envoie par exemple quatre messages d'incrémentation à un compteur (« Counter »), mais cette activité (composée de ces 4 messages) peut être interrompue en cas de réception de message « TurnLamp ».

Plus généralement, lorsqu'il entre dans un nouvel état, un objet effectue ses actions *On Entry*, puis une fois celles-ci terminées, enchaînent sur les actions *Activity* (notées « Do / ... »). Cependant, du point de vue de l'utilisateur, les objets effectuent les actions en parallèle les uns avec les autres (puisque la série d'actions *Activity* peut

être interrompue). Le but du mécanisme de séquencement est de simuler ce parallélisme en donnant la main aux objets qui effectuent ainsi leurs actions *Activity* chacun à leur tour.

Pour satisfaire à ces exigences, les modèles doivent être aménagés de façons particulières comme nous allons le voir dans la section suivante, ceci de manière transparente pour le modélisateur.

Section 2 - Aménagement des modèles

Mise sous forme canonique des objets

Pour simuler le parallélisme, le mécanisme de l'invention consiste à ordonnancer les actions *Activity* des objets à l'aide d'une entité particulière, à savoir un Séquenceur. Pour parvenir à cet ordonnancement, les états sont transformés en une forme que l'on appelle la « forme canonique », comme illustré en détail sur la figure 146, forme canonique dans laquelle ces états sont décomposés de telle sorte qu'à chaque action *Activity* corresponde un état. Le Séquenceur est alors chargé de piloter les transitions entre les états correspondant à des actions *Activity*.

Cependant, la transformation en forme canonique doit respecter la caractère atomique des actions *On Entry*.

Au global, la transformation en forme canonique doit donc respecter deux contraintes :

- préserver le caractère atomique des actions *On Entry* : Pour ce faire, les actions *On Entry* sont regroupées dans un état « Sentry », comme illustré sur la figure 145. Pour que ces actions ne soient pas interrompues, la seule transition sortante de cet état est une transition automatique. Une transition automatique est une transition à laquelle n'est associée aucun événement. Une telle transition n'a donc lieu qu'une fois toutes les actions de l'état Sentry effectuées.

Ainsi, lorsque la transition entrante est déclenchée (par un autre objet par exemple), l'objet entre dans l'état Sentry. Dans cet état, l'objet effectue toutes les actions *On Entry* de l'état S. Quand toutes les actions sont effectuées, et seulement quand elles le sont, l'objet entre dans l'état Sactivity. Comme dans l'exemple, l'objet n'effectuait pas d'actions *Activity* dans l'état S, il se place dans le sous-état Swait où il attend de recevoir un événement lui permettant de rentrer dans l'état T ou dans l'état U.

On notera ici que, dans la suite, afin d'apporter plus de clarté à la lecture, nous séparerons les actions *Entry* dans des sous-états différents.

- permettre d'établir un quasi parallélisme dans l'exécution des activités des objets. Plutôt que d'avoir un « thread » par objet, les activités de tous les objets peuvent être traitées dans un seul thread : celui du Séquenceur. Le rôle de ce Séquenceur sera donc de régir le passage d'un sous état d'activité à un autre par l'émission de l'événement *next*, pour tous les objets.

La mise sous forme canonique correspondante est illustrée sur la figure 146.

Comme précédemment, l'objet entre dans l'état Sentry et y effectue toute les actions *On Entry* prévues dans l'état S. Puis, l'objet entre automatiquement dans l'état Swait. Nous sommes ici dans le cas où des actions *Activity* étaient prévues dans l'état S. Chacune de ces actions est traitée dans un état différent. Nous les appellerons des sous états d'activité. La transition d'un sous état d'activité à un autre se fait à la réception de l'événement *next*. C'est le Séquenceur qui émet cet événement pour simuler le parallélisme entre les objets.

A tout moment, l'objet peut recevoir un événement associé à une des transitions sortantes. A ce moment là, l'objet sort de l'état Sactivity (et donc du sous-état d'activité dans lequel il se trouve) pour entrer soit dans T soit dans U.

Rôle de Swait

Les transitions sortantes de l'état S partent, dans la forme canonique, de l'état Sactivity. En effet, si elles partaient également de Sentry, les actions *On Entry* pourraient être interrompue, contre-disant leur caractère atomique. C'est pourquoi une fois les actions *On Entry* terminées, l'objet passe automatiquement dans le sous-état Swait de Sactivity. Dans cet état, l'objet attend de recevoir le premier next du Séquenceur, mais peut également passer à un autre état si un objet déclenche une des transitions sortantes.

On observera ici que la forme canonique décrite ci-dessus en référence à la figure 146 permet de représenter les diagrammes d'états transitions de Harel (états hiérarchiques et états orthogonaux) ; les transformations de ces cas étant évidentes pour l'homme du métier, elles ne seront pas décrites plus avant.

Notion d'historique (trace)

Afin de pouvoir suivre ce qui se passe dans le système informatique, on peut disposer d'un historique qui relève toutes les actions effectuées par les objets et leur associe un temps logique.

Avant d'effectuer une action, un objet en informe le séquenceur par un message approprié, et le séquenceur écrit dans la trace les informations suivantes :

- Quel objet effectue la transition
- Quel était l'état de l'objet avant la transition (on l'appellera état précédent), au moyen d'un identifiant.
- Quelle est la « cause » de cette transition (soit l'utilisateur *user* le cas échéant, soit l'objet qui a causé la transition)

On notera que les conditions spécifiées sur les transitions (voir UML – Unified Modeling Language, déjà mentionné) ou dans les états, peuvent également être notées dans la trace.

Cet historique va jouer un rôle dans le procédé de virtualisation décrit dans le chapitre suivant, ainsi que dans le mécanisme de Rollback explicité dans le chapitre VI.

Considérons l'exemple d'un objet O1 illustré sur la figure 147. La mise sous forme canonique va donner le résultat illustré sur la figure 148 ; l'appel à *put* du Séquenceur, illustrée sur la figure 148, correspond à une écriture dans l'historique avec les éléments que nous avons mentionnés plus haut.

Supposons maintenant que l'utilisateur crée cet objet O1. Celui-ci va donc entrer dans l'état Sentry, et effectuer son action Entry1. Au passage, la trace va être renseignée de la façon illustrée sur la figure 149.

Dans l'appel à méthode « Put(O1, initial, user) », « O1 » désigne l'objet qui effectue la transition, « initial » désigne l'état précédent de l'objet, tandis que « user » désigne ici l'objet (en l'espèce une action de l'utilisateur) qui est à l'origine de la transition (que l'on appellera la « cause »). « 54 » désigne ici le temps logique auquel le message est reçu par le séquenceur.

Le curseur d'écriture indique où doit se faire la prochaine entrée dans la trace.

Le Séquenceur doit donc être capable de réagir à l'événement *put*, en écrivant les différentes informations reçues dans l'historique. Le comportement du séquenceur, en ce qui concerne la gestion de l'historique, peut alors être représenté de la façon illustrée sur la figure 150, où H représente Historique.

Exclusion mutuelle des premiers cycles

- Notion de cycle

Lorsqu'un objet déclenche un événement chez un autre (en lui envoyant un message ou par une notification de changement d'état si l'objet y est abonné, etc), il provoque

un changement d'état de ce dernier. Dans son nouvel état, celui-ci peut lui-même faire changer d'état à un troisième objet, etc... Ceci jusqu'à ce que tous les objets touchés aient terminés leurs actions *On Entry* (les actions *Activity* étant du ressort du séquenceur comme nous le verrons par la suite). La main est alors rendue, par retour d'appel de méthode, à l'objet qui a déclenché la première transition : on parle alors de cycle. Par définition, un cycle est donc atomique au sens où il ne peut pas être interrompu.

Un cycle débute donc quand un objet O en amène un autre à effectuer une transition, et se termine quand O reprend la main.

Ainsi la figure 151 illustre un exemple où deux cycles Cycle1 et Cycle2 se succèdent, le retour à partir de l'objet ObjetC se produisant dès la réception de « EntryB1 » car, si ObjetB change bien d'état, il n'a pas d'action à effectuer. Le cycle Cycle2 est similaire, avec parcours des objets ObjetA, ObjetC et ObjetD et retour à partir d'ObjetD qui n'a pas d'action à effectuer.

- Notion de premier cycle

Lorsqu'on considère l'ensemble des cycles d'une application, on se rend compte qu'il existe des cycles tout particuliers : ceux qui sont initiés par l'utilisateur. On parlera dans ce cas là de *premiers cycles*.

Ces premiers cycles sont importants car leur déclenchement dépend de l'utilisateur. Leur déroulement est de ce fait parallèle et ils peuvent entrer en conflit les uns avec les autres si jamais ils passent par les mêmes objets.

Pour gérer ce genre de problème, c'est-à-dire éviter d'avoir à effectuer des retours arrière (ou « Rollback » selon la terminologie anglosaxonne), un mécanisme d'exclusion mutuelle de ces premiers cycles est nécessaire. Comme nous allons le voir dans la section suivante, une solution possible est de les mettre en séquence : ils

sont alors traités les uns à la suite des autres en fonction de l'ordre selon lequel ils ont été déclenchés par l'utilisateur.

- Mécanisme mis en œuvre

Dans cette approche, chaque objet qui effectue une transition due à l'utilisateur prévient le Séquenceur à l'aide de l'événement BOFC (BeginOfFirstCycle : début de premier cycle). Une fois le premier cycle terminé (on est revenu au premier objet par retour d'appel de méthode), l'objet prévient le Séquenceur que le premier cycle est terminé à l'aide de l'événement EOFC (EndOfFirstCycle : fin de premier cycle).

A la réception d'un message BOFC, un compteur de premier cycle FCC (First Cycle Counter), intégré dans le Séquenceur, est incrémenté. Si, après incrémentation, ce compteur vaut 1, alors le Séquenceur dit à l'objet que le premier cycle peut commencer en envoyant le message *proceed*. Quand le Séquenceur reçoit EOFC, il décrémente son compteur de premier cycle.

Lorsque le Séquenceur reçoit un BOFC alors qu'un premier cycle est déjà en cours, il attend que le cycle en cours (ainsi que tout ceux qui sont en attente) soit terminé, avant de déclencher le nouveau premier cycle.

Un exemple de mise en œuvre est illustré sur le diagramme de la figure 152.

Le comportement du Séquenceur relativement aux BOFC et EOFC peut être représenté à l'aide du diagramme d'état-transition illustré quant à lui sur la figure 153.

Obtention du quasi parallélisme

Après avoir effectué leurs actions On Entry, les objets peuvent effectuer leurs actions *Activity*. Par définition ces actions ont lieu en parallèle. Le but de cette partie est de

décrire le mécanisme de quasi parallélisme permettant de simuler le parallélisme des activités dans une approche purement séquentielle.

- Extension de l'historique

Comme nous l'avons vu, la décomposition en forme canonique dégage des sous-états d'activités. Le passage d'un de ces sous états à un autre se fait par l'événement *next*.

Ceci permet de piloter les différentes actions Activity de tous les objets qui effectuent leur activité. Ce pilotage est assuré par le Séquenceur, chargé d'émettre l'événement *next* aux objets en ayant fait la demande.

Pour ce faire, un objet va dire au séquenceur qu'il a une activité à effectuer juste avant la dernière action de sa partie *On Entry* de l'état courant, en l'écrivant dans l'historique. (Au cas où il n'a qu'une seule action *On Entry* dans l'état courant, l'objet va informer le séquenceur avant d'effectuer l'action). Ceci est effectué via l'écriture dans l'historique à l'aide de *put*. L'ensemble d'informations stockées dans l'historique à chaque *put* doit donc être étendu de la façon suivante : en plus des informations déjà présentées, un *put* doit également contenir un *type* et un *index* d'obsolescence.

Le *type* indique s'il s'agit juste d'une simple écriture dans l'historique (type = trace), ou s'il s'agit d'une demande pour effectuer des activités (type = next).

L'*index* d'obsolescence (renseigné uniquement dans le cas type = next) permet de détecter si un *next* reçu est caduque suite à un changement d'état pendant des activités.

Si nous reprenons notre exemple de mise sous forme canonique de la figure 148, nous obtenons maintenant ce qui est illustré sur la figure 154, et dans le cas où l'objet doit effectuer des actions *Activity*, ce qui est illustré sur la figure 155.

Supposons qu'un utilisateur crée cet objet (dans le cas des actions *Activity*). Les informations figurant dans la trace pour cet objet vont donc être par exemple celles illustrées sur la figure 156.

Le diagramme d'état-transition présentant le comportement du Séquenceur pour la gestion de l'historique doit alors être étendu comme illustré sur la figure 157.

- Rôle du Séquenceur

Le rôle du Séquenceur est de régir les différentes activités des objets en simulant une exécution en parallèle.

Pour ce faire, le Séquenceur exploite les informations de l'historique en suivant les différentes entrées, et en traitant les *put* dont type = next. Ceci est matérialisé par un autre curseur, le curseur de traitement, comme illustré sur la figure 158. Ce curseur indique donc, à chaque instant, où en est l'exécution de l'application.

A chaque fois que le curseur de traitement passe sur un *put* de type next, le Séquenceur envoie *next* à l'objet spécifié dans le put.

Le traitement des activités par le Séquenceur peut donc être modélisé par le diagramme d'état-transition illustré sur la figure 159.

Ainsi le diagramme d'état-transition complet du Séquenceur sera donc tel qu'illustré sur la figure 160.

Exemple

Considérons 6 objets A, B, C, D, E, et F. Ces objets ont les diagrammes d'état-transition qui sont illustrés sur la figure 161.

A, B, et C sont mis sous forme canonique de la façon illustrée sur la figure 162.

Supposons qu'un utilisateur déclenche le msg0 de l'objet A. A entre donc dans l'état S1, il en découle un premier cycle, puis le Séquenceur arbitre les différentes activités. Ceci va être présenté dans le diagramme de séquence illustré sur la figure 163.

Au moment où le premier cycle se termine, l'historique porte les informations illustrées sur la figure 164.

Le séquenceur va commencer à traiter les activités à partir de la position du curseur du Séquenceur. Pour chaque *put* de type next, il va envoyer à l'objet concerné le message *next* pour que celui-ci puisse continuer son activité. Si on poursuit l'exemple précédent jusqu'à son terme, on obtient le diagramme illustré sur la figure 165.

Lorsque le traitement du premier *put* de type next est terminé, le curseur de traitement passe au put dont type = next suivant, comme illustré sur la figure 166, et le traitement correspondant est illustré sur la figure 167.

Au final, la trace regroupe les informations illustrées sur la figure 168.

- Obsolescence des « next »

Les activités d'un objet peuvent être interrompues par un changement d'état. On dira que le Séquenceur envoie un événement *next* obsolète si l'objet a fait au moins une

transition depuis qu'il a émis le *put* de type next correspondant. Ceci peut arriver lorsque les activités propres à un état sont interrompues au sens où nous l'avons vu précédemment.

Pour éviter ce problème, chaque objet dispose d'un compteur des émissions de *put* de type *next* (initialisé à zéro en début d'exécution). A chaque fois qu'un objet émet un *put* (de type *trace* ou *next*), il incrémente auparavant ce compteur et, dans le cas d'un *put* de type *next*, passe la valeur du compteur dans l'historique. Accessoirement, chaque objet possède également un drapeau *PendingNext* permettant de savoir s'il est en attente ou non d'un *next*. A l'envoi d'un *put* de type *next*, ce drapeau passe à 1 (il est remis à 0 à la réception du *next*).

Par la suite, quand l'objet reçoit un *next* et il reçoit également la valeur qu'il avait passé au moment de l'envoi du *put* de type *next*. L'objet compare alors cette valeur qu'il reçoit avec la valeur courante de son compteur. Si les deux valeurs sont égales, c'est que le *next* reçu doit être traité, sinon c'est qu'il est obsolète et est donc ignoré.

Un exemple correspondant est illustré sur la figure 169.

* * *

Chapitre V - Mécanisme de chargement et déchargement progressifs (Figures 170 à 196)

Comme le chapitre IV, ce chapitre contribue au chapitre VI présenté plus loin, et vient compléter les techniques dudit chapitre IV.

On se place ici dans le cadre d'une architecture informatique client/serveur, où un utilisateur exécute une application depuis un poste client « léger », dans le sens où les objets de l'application sont instanciés depuis un serveur. En fait, nous distinguons deux types d'objets : les objets transitoires, et les objets permanents. Les objets

transitoires sont propres à chaque exécution d'un document ; ils ont une durée de vie qui ne peut en aucun cas excéder celle de l'exécution de l'application. L'état et les valeurs caractéristiques des objets permanents sont, au contraire, sauvegardés (sachant que l'on est ici dans un cas où cette sauvegarde est effectuée sur un serveur distant) et survivent d'une exécution à l'autre.

De par la structure induite par le diagramme de classe, un certain nombre d'objets « existent » d'emblée au lancement de l'application : le modèle (au moyen duquel le document a été créé) indique que, dès le début de l'exécution, ces objets existent déjà et sont à disposition les uns des autres (en fonction des associations définies dans le modèle). Le principe de base de ce chapitre est cependant de ne pas créer ces objets dès le début, mais de manière « paresseuse » (au fur et à mesure de leur besoin), et ceci de manière transparente pour l'utilisateur. Le concepteur n'a en effet pas besoin de prévoir ces aspects lors de la modélisation de l'application à développer ; ils sont automatiquement gérés par le système. L'utilisateur, quant à lui, n'a pas à se soucier des objets qui transitent entre son poste client et le serveur, tout se passe pour lui comme s'il possédait l'ensemble des objets de l'application.

Ainsi, certains objets sont en fait absents quand, selon le modèle, ils devraient exister ; on dit qu'ils existent *virtuellement*. Le comportement de l'application n'en est pas gêné car ces objets sont créés (*réalisés*) « just in time » en cas de besoin. Ils sont même réalisés par *anticipation* (« ahead of time ») afin d'assurer la fluidité de l'exécution. Même en cours d'exécution, selon le modèle de nouveaux objets apparaissent virtuellement et, toujours, ils ne sont réalisés qu'au fur et à mesure de leurs besoins réels.

De ce fait, le téléchargement des objets se fait progressivement. Cela permet de conserver des temps de chargement raisonnables dans des conditions de bande passante faible et irrégulière.

Afin de réduire la consommation de ressources d'ordinateur, les objets qui ne sont pas (ou plus) utilisés peuvent cesser d'être réellement existants (ils peuvent être *virtualisés*). Ils sont supprimés, de manière transparente, après un certain délai de latence (pour éviter des suppressions/créations intempestives), sachant qu'ils pourront se réaliser à nouveau quand nécessaire.

Section 1 - Structure d'une application

Liens

La *structure* d'une application est basée sur le diagramme des classes et exploite les associations entre les classes. En cours d'exécution, la structure repose donc sur les liens entre les objets. Un *lien* est une instance d'association orientée. Parmi les différentes associations on distingue la *composition* (relation contenant-contenu) qui implique qu'un objet ne peut être contenu que par un seul objet (l'objet contenant) à un instant donné. Les associations de composition sont notées graphiquement sous la forme de losanges pleins. Par exemple, dans la figure 170, l'objet A contient les objets B, C et D.

Une application contient donc un ensemble d'objets, dotés de comportement. Ces comportements (ou *behaviors*) sont définis à l'aide de machines à états telles que nous les avons vues à la section 1 du chapitre IV.

Les objets forment une structure *arborescente* de *liens de composition*, au sein de laquelle tout objet est nécessairement composant d'un autre objet. Tout objet a donc un « chemin » qui l'identifie depuis la racine de l'application, l'objet Root, comme illustré sur la figure 171.

Les techniques décrites ici peuvent s'appliquer à des structures dynamiques, c'est-à-dire dans lesquelles certains objets peuvent être créés et supprimés en cours d'exécution. Nous considérons que ces derniers sont regroupés au sein de ce que

nous appelons une collection (les procédés décrits ici pouvant tout aussi bien s'appliquer à n'importe quelle autre structure dynamique). La collection est représentée par un objet Collection qui est statique, et qui figure dans l'arborescence basée sur les liens de composition. Les sous objets de l'objet Collection représentent les éléments de la collection comme le montre la figure 172, et sont repérés par des indices.

Bien entendu, les objets peuvent aussi avoir entre eux d'autres liens que les liens de composition qui forment la structure de l'application. Ces liens sont les instanciations des différentes associations existant entre les classes des objets. C'est via ces liens que les objets peuvent communiquer, c'est-à-dire s'envoyer des messages.

Ces liens peuvent évoluer en cours d'exécution. En effet, une association relie deux classes. Ceci sous-entend que l'instance d'une classe peut communiquer avec n'importe quelle instance de l'autre classe.

La communication entre un objet quelconque et un élément d'une collection (par exemple entre A et B[2] dans la figure précédente) se fera par l'intermédiaire de l'objet Collection correspondant, en précisant l'indice de l'élément voulu (ici 2).

Un lien entre deux objets est défini par un chemin menant de l'un à l'autre, au sein de la structure.

Un lien peut être construit en exploitant uniquement les liens de composition, nous les appellerons « liens de base ». La figure 173 en montre un exemple. Sur cette figure, le chemin menant de D à C est formé à partir des liens de composition D_B, B_A et A_C. Un tel chemin (définissant un lien de base) est porté par l'objet au sein d'une variable structurée que nous appellerons « port ».

Mais un lien peut également exploiter des liens qui ont eux-mêmes été formés à partir de liens de composition, c'est ce que nous appellerons un « lien dérivé ». Nous en illustrons un exemple dans la figure 174.

Le chemin menant de D à E peut réutiliser le chemin établi entre D et C dans l'exemple précédent : D_C et C_E. Il n'est pas nécessaire de passer par B et par A, mais il est possible de déterminer le chemin correspondant qui n'exploite que les liens de composition. Ceci jouera un rôle majeur, comme nous le verrons dans la première partie de la section 2 du chapitre V, en ce qui concerne la réalisation des objets.

On peut donc définir un lien de façon récursive, comme étant soit :

- Un lien de base, c'est-à-dire n'exploitant que des liens de composition
- Un lien dérivé, c'est-à-dire exploitant un nombre fini de liens (pouvant être de base et/ou dérivés).

Objets d'affichage

Certains des objets d'une application sont des éléments d'affichage, tels qu'un bouton ou une zone de saisie par exemple. Les éléments d'affichage sont regroupés pour former différents écrans de l'interface homme machine d'une application, différentes pages d'un site Internet, ou différents champs de vision dans un environnement de réalité virtuelle en 3D par exemple.

Ce regroupement des éléments d'affichage est réalisé à l'aide d'objets, créés spécialement, que nous appelons *visuZone*. Chaque groupe d'éléments d'affichage est lié à un objet *visuZone*. Ces groupes ne sont pas forcément disjoints. Ainsi la figure 175 illustre un groupe d'objets d'affichages D, F et H regroupés.

Regrouper de cette manière les objets d'affichage a le but suivant : en cours d'exécution, le chargement chez un client d'un objet d'affichage appartenant à une

zone de visualisation, provoque le chargement de tous les autres objets d'affichage de la même zone. Selon la profondeur d'anticipation désirée, les objets (d'affichage ou non) reliés aux objets de la zone de visualisation peuvent également être chargés.

Un exemple intéressant d'utilisation desobjets *visuZone* peut être le déplacement d'un cadre sur une carte. Les objets *visuZone* permettent de gérer des concepts comme le voisinage géographique et le zoom.

- Voisinage géographique

On se place ici dans un contexte où l'application contient un nombre important de données, réparties de façon géographique sur un plan (par exemple la carte d'une ville). La navigation sur cette carte se fait au moyen d'un cadre de visualisation constitué par la surface de l'écran de l'utilisateur.

Le but de l'utilisation des objets *visuZone* est ici de faire en sorte que seuls les objets du cadre et ceux qui en sont géographiquement proches soient présents.

Pour réaliser cet objectif, il faut dans un premier temps morceler la zone en 2D à couvrir, et associer à chaque morceau un objet *visuZone* (c'est-à-dire relier tous les objets de ce morceau de plan à l'objet visuZone). Ceci est illustré sur la figure 176, où à chaque « morceau » du rectangle est associé un objet *visuZone* regroupant les objets, désignés par des points, qui y sont contenus.

Dans un second temps, il faut inclure la notion de proximité entre les zones de visualisation : il s'agit donc de relier les objets *visuZone* proches les unes des autres. On obtient ainsi un réseau maillé d'objets *visuZone*, comme illustré sur la figure 177, où les objets *visuZone* sont désignés par des cercles au droit de chaque « morceau », et où les liens entre de tels objets sont indiqués par des lignes droites.

Ainsi, le chargement d'un objet *visuZone* provoque (automatiquement), par anticipation, le chargement des objets d'affichage situés dans les zones voisines.

De cette manière, la profondeur d'anticipation et la taille des zones de visualisation sont les paramètres qui permettent de jouer sur la granularité du « maillage ».

- Zoom avant et arrière

Dans le type d'application que nous décrivons dans cet exemple, une fonctionnalité intéressante est également la possibilité d'effectuer des zooms avant et arrière en un point donné de la carte.

Cette fonction induit le concept de niveau de détail, donc de hiérarchie au sein des objets *visuZone*, un exemple d'une telle hiérarchie étant illustré sur la figure 178. L'objet *visuZone* supérieur correspond au niveau de détail moins élevé, alors que les objets *visuZone* situés au-dessous représentent un niveau de détail médian, et que les objets *visuZone* les plus bas représentent un niveau de détail plus élevé.

Chaque objet *visuZone* du niveau de détail le plus élevé est relié à tous les objets d'affichage de la zone de visualisation de granularité la plus fine.

L'objet *visuZone* du niveau de détail le moins élevé (objet supérieur) est relié à certains objets de toutes les parties du plan morcelé. Ceci constitue une vue globale de la carte où seuls les éléments les plus marquant apparaissent.

Les objets *visuZone* du niveau de détail médian (objets intermédiaires) sont reliés à des objets supplémentaires de leur quart de plan respectif. Ces objets apportent plus de précision sur chaque quart de plan.

Enfin les objets *visuZone* du niveau de détail le plus élevé (objets inférieurs) sont reliés à tous les objets restant dans leur fraction de plan. A ce niveau, tous les objets sont chargés.

De même que pour la question de proximité, les objets *visuZone* correspondant aux différents niveaux de détail sont reliés entre eux dans un souci d'anticipation.

Une approche de zoom sans niveau de détail peut également être envisagée. Dans celle-ci, les objets *visuZone* inférieurs sont reliés à tous les objets de leur fraction de plan. Les objets *visuZone* intermédiaires ne sont reliés qu'aux objets inférieurs, etc...

Section 2 - Modalités de Réalisation et de Virtualisation des objets

Lors du lancement de l'application, seule la racine du document (objet Root) est réalisée. Elle constitue la racine de toute l'arborescence des objets présents dans le document. A partir de là, les autres objets sont « réalisés » au fur et à mesure, sur demande de la part d'un autre objet, ou par anticipation.

Ainsi, lorsqu'un objet doit envoyer un message à un autre objet non présent, la réalisation de ce dernier est effectuée au cours du processus d'envoi de message.

Ce processus est intercepté par une couche spéciale du système qui fait que les différents mécanismes mis en jeu dans ce chapitre (réalisation, virtualisation, anticipation) soient gérés de façon transparente pour le concepteur et l'utilisateur.

Dans toute application orientée objet, pour qu'un objet puisse envoyer un message à un autre objet, il faut qu'il existe entre eux un *lien*, au sens où nous l'avons défini précédemment. Ce lien est créé à partir du moment où l'objet émetteur connaît le chemin qui mène à l'objet destinataire au sein de la structure. Un tel lien n'a pas de caractère figé, il peut être « modifé » en cours d'exécution, en en modifiant le chemin.

Le rôle du chemin est de permettre, en parcourant la structure, de récupérer la référence de l'objet destinataire pour le contacter. L'obtention de cette référence est le but d'un mécanisme que nous appellerons procédé de connexion (défini ci-après) qui fait partie intégrante du processus d'envoi de message. Cette référence, une fois obtenue, est stockée par l'objet au sein de la variable structurée « port » qui contient déjà le chemin.

Ce procédé ne peut fonctionner que pour des chemins formés de liens de composition uniquement. En effet, ce n'est qu'en suivant les liens de composition que l'on peut créer les instances des objets (un objet ne peut être *réalisé* que par son parent). Atteindre un objet en suivant un chemin implique de pouvoir réaliser les objets de ce chemin qui ne sont que virtuellement présents, d'où la nécessité qu'un tel chemin repose sur les liens de composition.

Pour cela, il est possible de ramener un lien quelconque au lien de base correspondant, c'est-à-dire à l'ensemble minimal de liens de composition qui le constituent.

Les deux cas possibles de transformation sont présentés sur la figure 179: dans les deux cas illustrés, la composition des liens de O_0 à O_n, et de O_n à O_m, peut se ramener au lien de base formé des liens de composition $O_0\_O_i$, $O_i\_O_j$, et $O_j\_O_m$.

Procédé de connexion (réalisation)

Supposons qu'un objet O_0 veuille se connecter à un objet O_N. Afin d'accéder à O_N, O_0 dispose d'un chemin path qui exploite la structure arborescente. Une valeur de ce chemin peut être stockée dans le port, mais cette valeur peut également être modifiée de façon programmatique en cours d'exécution, sachant qu'un port (lien de base) ne peut contenir que des liens de composition.

Pour représenter un lien de base, un port comprend une valeur entière index positive et une suite d'objets $(O_n)_{n\in[i..N]}$. index indique à quel niveau de la hiérarchie, à partir de l'objet qui se connecte, se situe l'ancêtre commun O_i. La suite $(O_n)_{n\in[i,N]}$ regroupe les objets qui mènent de O_i à O_N, elle constitue le chemin « descendant » (vers O_N), que nous noterons descendingPath.

Chaque O_n peut désigner soit un objet soit un élément de collection. Dans ce dernier cas, on notera $O_n[p]$ où O_n est l'*objet* collection qui regroupe tous les *éléments* de collection, et où p représente l'indice de l'élément au sein de la collection.

La structure UML correspondante est illustrée sur la figure 180.

La réalisation se faisant de parent à enfant (un objet ne peut être réalisé que par son unique parent au sens de la relation de composition), l'existence de O_0 implique que tous les objets, jusqu'à l'ancêtre commun O_i, ont été réalisés au moins une fois (même s'ils ont pu être virtualisés depuis).

Le procédé de connexion réside en l'appel récursif d'une méthode *connect* portée par chaque objet. Cette méthode retourne la référence de l'objet auquel on désire se connecter. Elle prend comme argument le chemin *(index,descendingPath)* menant à l'objet auquel on veut se connecter et l'argument *depth* qui indique à quelle profondeur vont être effectuées les connexions, comme nous allons le voir dans la section 3 du chapitre V. Ce procédé est lancé en appelant la méthode symbolisée par :

$$O_0.connect(index,descendingPath,depth)$$

Une fois la référence de O_N récupérée, O_0 envoie sa propre référence que O_N stocke dans un attribut *inverseRef*. En effet, si O_N est virtualisé, la référence que possède O_0 n'a plus de sens. O_N doit donc prévenir O_0 qu'il est virtualisé en mettant à Null sa

référence portée par O_0. Il peut le faire grâce à *inverseRef*. O_0 sait alors qu'il devra se reconnecter à O_N pour lui envoyer un message.

Si le parent d'un objet (qui désire se connecter) a été virtualisé, il est réalisé de nouveau au cours du procédé de connexion. Pour ce faire, chaque objet possède la référence de l'objet *Root*. Lorsqu'un objet détecte que son parent a été virtualisé (la référence de son parent est à Null), il passe à l'objet *Root* le chemin qui conduit de la racine à son parent. L'objet *Root* peut alors instancier le parent en s'y connectant avec le chemin qu'il a reçu. Une illustration de ce phénomène est donnée dans l'exemple montré à la figure 181.

Sur cette figure, supposons que A veuille envoyer un message à E en utilisant le lien défini par le chemin (3,myC_myE). A ne peut accéder à son parent D car ce dernier a été virtualisé (en gris sur la figure), A n'a donc plus la référence de D. Pour l'obtenir de nouveau, il lui faut passer par l'objet *Root*.

Maintenant en référence à la figure 182, A connaît son propre chemin depuis la racine (myB_myD_myA). Il peut donc inférer celui de son parent D : myB_myD. Ce chemin est passé à l'objet Root qui va alors procéder à l'instanciation des objets manquant jusqu'à D (ici B à l'étape 1 et D à l'étape 2).). Une fois D instancié, sa référence est passée à A par retour de méthode.

Le procédé de connexion peut alors se résumer comme suit :

- Parcours de l'arbre du modèle jusqu'à l'objet visé en passant par un ancêtre commun.
- Réalisation « au passage » des objets ayant été virtualisés, ou n'ayant pas encore été réalisés.
- Récupération de la référence de l'objet visé, qui est stockée dans la variable port.
- Envoi de la référence de l'objet qui se connecte à l'objet visé (inverseRef).

Prenons pour exemple le cas où, au lancement de l'application, l'objet Root envoie un message à l'objet visuZone qui regroupe les éléments d'affichage qui constituent la page d'accueil de l'application. On a le chemin illustré sur la figure 183, où l'objet *Root* est blanc, l'objet *visuZone* est noir et les objets d'affichage sont gris.

A la réception de ce message, l'objet *visuZone* se connecte à ses éléments d'affichage, exactement selon le même procédé décrit ci-dessus. Ceci est détaillé dans la figure 184, et peut se traduire par le diagramme d'état-transition de la figure 185.

Cas de liens de composition non immuables

Optionnellement, en cours d'exécution, il se peut que des liens de composition soient modifiés. Le fait qu'un objet n'ait qu'un seul parent à un instant donné n'empêche pas d'avoir plusieurs parents à des instants différents.

Considérons l'arborescence illustrée sur la figure 186.

Sur cette figure, les traits pleins représentent les liens de composition et les traits pointillés représentent les liens d'associations simples. Afin d'obtenir la référence de J, F dispose d'un chemin (1, G_I_J). La valeur 1 indique qu'il faut remonter d'un cran dans la hiérarchie des ancêtres de F pour trouver l'ancêtre commun à F et I (en l'occurrence D). Le chemin G_I_J montre qu'il faut passer par G et I pour atteindre J. Si I est virtualisé, c'est à G de le réaliser de nouveau, de même entre J et I.

Supposons maintenant que le lien de composition entre G et I soit rompu, et remplacé par un lien entre H et I, comme le montre la figure 187.

Si J est virtualisé, F perd la référence qu'il avait préalablement acquise. Et dans le cas où F doit envoyer un autre message à J, il doit déterminer à nouveau cette

référence grâce au chemin qu'il porte. Or, le parent de I ayant changé, le chemin en question n'a plus aucun sens.

Pour pallier à ce problème, nous introduisons une indirection. La variable structurée *port* doit contenir, pour les liens de composition de parent à enfant, non seulement la référence de l'enfant, mais également celle de l'objet lui-même ou du nouveau parent qui le remplace.

Si on reprend l'exemple de la figure 186, on obtient dans une première étape, comme illustré sur la figure 188, une référence vers le parent (flèche gris clair) et une référence vers le fils (flèche gris foncé). G est le parent de I, et la référence vers le parent qu'il porte est une référence vers lui-même.

Et dans la seconde étape où H devient le parent de I, comme illustré sur la figure 189, la référence du parent de I portée par G est maintenant celle de H, tandis que la référence de G sur son ex-enfant I est mise à Null du fait que G n'en est plus le parent.

La variable port de H prend alors la forme qu'avait celle de G dans l'étape précédente. La référence parent pointe sur H et lui même, et la référence enfant pointe sur I.

G porte maintenant la référence de H, mais également le chemin qui mène à H (3,myC_myE_myH). Si H est virtualisé, G peut ainsi le réaliser à nouveau grâce à ce chemin.

Ainsi, lorsque F veut de nouveau obtenir la référence de I, il peut utiliser le même chemin (1,G_I). Une indirection se fait alors automatiquement en suivant la référence parent de G vers H.

Virtualisation

Le fait qu'un objet peut être virtualisé pour économiser des ressources d'ordinateur, a été introduit en début du présent chapitre.

Comme nous l'avons aussi vu en introduction de ce chapitre, nous disposons d'objets transitoires (propre au poste client et qui ne survivent pas à l'exécution de l'application) et des objets permanents (sauvegardés sur un serveur). Un objet permanent peut être librement « virtualisé » sur le client (quand il n'est pas utilisé), puisque son état courant peut toujours être récupéré à partir du serveur. Par contre, un objet transitoire ne peut être virtualisé que quand il est dans son état initial. En effet, de par sa nature, son état n'est présent qu'en mémoire sur le poste client de l'utilisateur, et n'est sauvegardé nulle part. Il ne peut donc être réalisé qu'à son état initial. De fait, si un objet transitoire était virtualisé dans un autre état que son état initial, nous perdrions de l'information.

La virtualisation est déclenchée par les objets *visuZone* : lorsque des objets d'affichage disparaissent de l'écran (par exemple parce que l'utilisateur a changé de page), un temporisateur unique est déclenché. Si aucun élément d'affichage relié à l'objet *visuZone* n'est réaffiché avant l'expiration de la temporisation, tous les éléments sont virtualisés.

La virtualisation de ces éléments peut provoquer celles des objets qui y sont connectés (liés), ainsi que des objets qui sont connectés à ces derniers, et ainsi de suite. En effet, à l'aide de Séquenceur et de l'historique, il est possible de retrouver quels sont les objets qui manipulent les objets d'affichage en suivant les relations de cause à effet (dans le sens des effets aux causes).

Supposons maintenant, dans l'exemple illustré sur la figure 190, que F soit un objet d'affichage pouvant être virtualisé. L'historique nous indique qu'à l'instant logique t+7, E était à l'origine d'une transition de F, puis à l'instant t+5, D était à l'origine

d'une transition de E, ...et ainsi de suite jusqu'à A dont la transition était initiée par l'utilisateur. A, B, D, et E sont donc candidats à la virtualisation (virtualisables) quand F se virtualise.

Cependant, un objet candidat à virtualisation peut être en train d'interagir avec d'autres objets, et il ne faut alors pas le virtualiser (on notera ici qu'une interaction, proche dans le temps, prévue avec des objets non liés à l'objet *visuZone* actif courant ou à ses vosins, n'est pas un frein à la virtualisation si ces objets sont répliqués et vivent sur le serveur - voir plus loin chapitre VI). Pour pouvoir détecter ces cas de figure, on dispose de plusieurs techniques :

- Comme nous l'avons vu, chaque objet dispose d'un drapeau *PendingNext* qui indique si il est en attente d'un next pour poursuivre ses activités. Un objet dans ce cas ne doit bien sûr pas être virtualisé, puisqu'il serait réalisé à nouveau peu de temps après, dès qu'il recevrait le *next*.

- Un objet virtualisable (dans la chaîne de cause à effet qui a aboutit à la transition d'un objet d'affichage) peut être en train d'effectuer d'autres actions. Celles-ci sont alors reportées dans l'historique. Pour chaque objet virtualisable, il suffit de regarder dans l'historique s'il n'existe pas d'autres entrées le concernant (postérieures à celle figurant dans la chaîne de cause à effet).

- A chaque envoi de message, l'objet relève le temps CPU (temps de l'unité centrale de traitement qui l'exécute) dans un attribut *lastActionTime*. Lorsqu'un objet est déclaré potentiellement virtualisable (au sens où nous venons de le définir), on regarde de nouveau le temps CPU *currentTime*.
  - ✓ Si *currentTime* - *lastActionTime* > δ, δ étant un intervalle de temps prédéterminé, alors l'objet n'a pas interagi depuis longtemps et peut donc être virtualisé.

- ✓ Si *currentTime* - *lastActionTime* < δ, l'objet n'est pas virtualisé et la propagation s'arrête dans cette direction : le système ne cherche pas à virtualiser les objets qui lui sont connectés.

Des dispositifs complémentaires peuvent également être envisagés :

- Un système d'apprentissage qui va inférer les délais entre les différentes actions, ceci permettant d'éviter de virtualiser des objets juste avant qu'ils effectuent une action.
- Un mécanisme probabiliste permettant d'associer (de façon manuelle, ou par inférence de la part du système) à chaque objet une probabilité d'être virtualisé.

Section 3 - Mécanisme de réalisation anticipative

Deux grandes approches d'anticipation, non exclusives, peuvent être mises en œuvre : l'approche structurelle et l'approche comportementale.

L'approche structurelle, offre l'avantage d'être transparente (ne nécessite pas de spécification supplémentaire). Elle utilise les liens qui ont préalablement été établis entre les objets, en fonction de la structure des associations dans le modèle. Cette approche ne donne donc pas forcément beaucoup de résultats au lancement de l'application, mais est de plus en plus efficace au long de l'exécution.

Anticipation comportementale

L'approche comportementale consiste à exécuter le modèle de l'application avec un certain nombre de pas d'avance, grâce à des spécifications supplémentaires (des annotations) fournies explicitement dans le modèle. Il s'agit notamment d'ajouter dans le modèle des « transitions anticipatives » qui représentent les effets des impulsions probables que le système pourrait recevoir de son environnement extérieur.

Lors de l'exécution, les transitions anticipatives ne sont pas prises en compte comme de vraies transitions (autrement dit, une transition anticipative ne change pas l'état courant de l'objet mais change plutôt une -ou plusieurs- variable de travail dans laquelle est simulée l'état futur), elles servent néanmoins, notamment, à « réaliser » les objets qui seraient à l'état « virtuel ». (On observera ici que l'anticipation comportementale peut aussi servir au raisonnement temporel, pour l'aide à la décision tactique par exemple).

Ainsi, dans l'exemple illustré sur la figure 191, un objet *Switch* peut effectuer une transition de *Off* à *On* par le message *turnOn*. Pour que le système puisse automatiquement anticiper une transition de *Off* à *On*, ayant lieu trois secondes après l'entrée dans l'état *Off*, le modélisateur annote la transition anticipative (en pointillés) avec l'événement déclenchant *time-out(3s)* (bien entendu, il peut y ajouter des conditions, des actions, etc).

Le modélisateur déclare que les transitions anticipatives sont anticipées avec un temps d'avance δ (il peut déclarer ceci globalement pour toute l'application ou sélectivement). La transition prédictive de l'objet *Switch* se fait ainsi au bout de $3-\delta$ secondes si $\delta<3$, et immédiatement sinon, après l'entrée dans l'état *Off*.

Les transitions qui en découlent (les actions dans les états anticipés produisent des événements qui causent d'autres transitions) sont aussi anticipatives. Comme les premières, elles ne se font pas sur le même plan que l'exécution de l'application et servent à amplifier le processus de réalisation. Bien entendu, les états anticipés par le système n'ont aucun effet de bord (notamment sur l'interface homme machine). Ainsi, dans l'exemple précédent, si on suppose que l'objet *Switch*, dans son état *On*, déclenche une transition d'un objet *Lamp* qui serait à l'état « virtuel », l'approche comportementale, par anticipation de la transition *turnOn*, permet de « réaliser » l'objet *Lamp* avant que l'utilisateur déclenche (directement ou indirectement) la transition *turnOn* réelle.

Dans la suite, nous décrivons l'approche structurelle, même si nous retiendrons que l'anticipation comportementale en constitue un complément important, puisqu'elle permet d'amplifier les avantages de l'anticipation structurelle (les deux approches ont une relation synergétique).

Anticipation structurelle

Comme nous l'avons vu dans la première partie de la section 2 du chapitre V, un argument *depth* (profondeur) est passé lors des appels à la méthode *connect*. Cet argument permet d'établir le degré d'anticipation que l'on désire atteindre. En effet, une fois la connexion de O_0 à O_N terminée, si *depth*>0, O_N établira ses propres connexions (à la profondeur souhaitée), comme le montre le schéma de la figure 192, qui illustre une connexion à la profondeur 3 (*depth* = 3).

Considérons maintenant le diagramme de classe illustré sur la figure 193, et supposons que les réalisations se font à la profondeur 1. Alors la connexion de l'objet *visuZone1* à l'objet B va provoquer non seulement la création de B, mais aussi celle de l'objet D via la connexion de B à D.

Ceci est explicité dans le diagramme de séquence de la figure 194.

Cas des objets visuZones

Dans le cas où l'objet visé serait un élément d'affichage, l'anticipation se fait par l'intermédiaire de l'objet *visuZone*. Il assure le lien entre l'objet visé (le premier élément d'affichage) et les autres éléments du même groupe. Mais cet intermédiaire doit être transparent et ne doit donc pas intervenir dans le calcul de la profondeur d'anticipation.

Ainsi, dans la représentation de la figure 195 qui illustre une connexion à la profondeur 2, on observe que la connexion à un objet *visuZone* ne compte pas dans le calcul de la profondeur d'anticipation.

De plus, comme nous l'avons vu dans l'exemple du voisinage géographique, les objets *visuZone* peuvent être reliés les uns avec les autres. Ceci ne change en rien le mécanisme d'anticipation, si ce n'est que dans ce cas la connexion entre objets *visuZone* compte dans le calcul de la profondeur.

Ainsi la figure 196 montre, toujours dans le cas d'un connexion à la profondeur 2, que la connexion entre deux objets *visuZone* compte dans le calcul de la profondeur d'anticipation.

* * *

Chapitre VI - Partage inter clients (Figures 197 à 238)

Introduction

On va traiter ici du partage d'une ou plusieurs applications par plusieurs utilisateurs distants. On se place donc dans une architecture informatique client/serveur, où chaque client exécute une application dont certains objets peuvent être partagés par d'autres clients. Le partage pourra notamment être mis en œuvre en conséquence des suggestions/acceptations de contenus décrites dans les chapitres II et III.

Section 1 - Types d'objets

Nous avons vu précédemment, dans le cadre du mécanisme de chargement progressif décrit dans le chapitre V, qu'il existe deux types d'objets : les objets transitoires et les objets permanents. Dans le chapitre V, cette dernière catégorie est subdivisée en deux : les objets permanents privés et les objets permanents partagés.

Un objet permanent partagé peut être utilisé conjointement par tous les clients. Chaque utilisateur partage avec les autres les valeurs caractéristiques de l'objet (état, rôles, etc...). Lorsqu'un utilisateur modifie l'état d'un tel objet, cela est ressenti chez tous les clients qui ont réalisé (au sens du mécanisme de chargement progressif) cet objet.

Les objets permanents privés sont, comme leur nom l'indique, propres à chaque client. Les clients utilisent certes les mêmes objets au sens où ils utilisent la même application, mais les objets permanents privés sont décorrélés entre eux d'un client à l'autre.

La gestion du partage d'une ou plusieurs applications entre plusieurs clients revient donc à gérer les objets permanents partagés, de telle façon que la modification d'un tel objet par un utilisateur soit reportée chez tous les clients concernés, c'est à dire ceux chez qui cet objet existe réellement.

Section 2 - Architecture

Plusieurs approches peuvent être suivies pour gérer le partage inter-clients. La première repose sur une réplication par « valeurs » des données partagées. En référence à la figure 197, la réplication entre les différents clients PC peut alors être gérée au moyen d'un Système de Gestion de Bases de Données sur un ou plusieurs serveurs communs PS. Un tel système assure la permanence des informations partagées et s'occupe des accès concurrents aux données.

Dans un tel contexte, la gestion du déterminisme n'est pas nécessaire. En effet, le comportement des objets (les actions qu'ils effectuent) a lieu chez les différents clients et en aucun cas sur serveur. Ce dernier ne conservant que les différentes valeurs (état...) des objets partagés, il n'a pas besoin de reproduire le comportement des objets. A chaque modification d'un objet partagé O chez un client, les valeurs de

O sont mises à jour dans la structure de données, et reproduites chez les clients qui utilisent également O.

Si cette approche présente l'indéniable avantage de la simplicité, elle ne permet cependant pas la persistance des traitements. A partir du moment où aucun client ne travaille, les objets de l'application sont inertes puisqu'ils n'ont aucune vie sur le serveur. La persistance des traitements est surtout utile dans le cas d'applications de simulation où l'on ne voufrait pas nécessairement laisser le poste client allumé pour permettre la poursuite de la simulation.

Permettre cette persistance de traitement fait l'objet de la deuxième approche que nous allons détailler dans ce chapitre. Elle implique que les applications utilisées conjointement par les clients s'exécutent également sur le serveur. Ceci permet, comme nous allons le voir dans le chapitre suivant, un mécanisme de réplication par calcul. Ce mécanisme requiert par contre d'assurer un haut degré de déterminisme du comportement des objets partagés et d'avoir à disposition un mécanisme de rollback pour rattraper le non-déterminisme entre client et serveur (exécution optimiste sur le serveur, voir section 5 du chapitre VI). Les applications sur le serveur disposent également d'un Séquenceur régulant les objets partagés.

Section 3 - Mécanisme de réplication

Comme nous l'avons vu, des utilisateurs d'une application peuvent partager certaines instances d'objets permanents partagés. Les utilisateurs peuvent interagir avec ces objets, mais chacun à leur tour grâce à un mécanisme d'exclusion mutuelle explicité plus loin. Quand un utilisateur modifie un objet, étant donné que la même instance peut être partagée avec certains autres utilisateurs, ces derniers doivent être notifiés de cette modification. Ainsi la figure 198 illustre le cas où deux utilisateurs *User1* et *User2* manipulent différents objets (objets noirs pour User1, objets blancs pour User2).

Cependant, du fait du mécanisme de chargement progressif, des utilisateurs qui partagent des mêmes instances n'ont pas forcément tous réellement ces instances sur leur poste. Ainsi, quand un objet partagé est modifié, il n'est pas nécessaire de répliquer cette modification chez tous les clients qui partagent cette instance, mais uniquement chez ceux pour qui l'objet en question est réél (a été réalisé, et depuis n'a pas été virtualisé).

Toute modification d'un objet permanent partagé due à un message provenant d'un objet transitoire ou d'un objet permanent privé est aussi effectuée sur un serveur, par la réplication de la transition qui en est à l'origine (les actions effectuées par l'objet permanent partagé sont donc exécutées à la fois chez le client qui a verrouillé cet objet, et sur le serveur). Ces modifications sont ensuite reportées du serveur chez chacun des clients qui possède réellement cet objet. Ceci constitue le premier aspect de ce mécanisme de réplication : la réplication par « valeur ».

Si une des actions de l'objet permanent partagé provoque une transition d'un autre objet permanent partagé, celle-ci n'est pas répliquée du client (qui a le verrou) sur le serveur. Elle est directement « calculée » par le serveur. Bien sûr, la modification de ce second objet permanent partagé est aussi reportée chez chacun des clients qui le possède réellement. Ceci constitue le deuxième aspect de ce mécanisme de réplication : la réplication par « calcul ».

Un tel type de réplication n'est possible que si l'application partagée s'exécute également sur le serveur, en accord avec le modèle d'exécution spécifié dans le chapitre IV (donc régulé par un Séquenceur).

Ainsi, on dispose au global d'un mécanisme de « réplication mixte » :

- Par valeur : au sens où les transitions d'objets permanents partagés sont reproduites quand elles ont été causées par des objets purement locaux dont le serveur n'a pas la visibilité (transitoires et permanents privés)

- Par calcul : au sens où les transitions d'objets permanents partagés sont déduites par le serveur quand elles ont été causées par un autre objet permanent partagé (dont le serveur a alors la visibilité).

Ainsi la figure 199 illustre les deux types de réplication entre un client et un serveur de partage, lorsque sont mis en œuvre des objets transitoires, des objets permanents privés et des objets permanents partagés.

Bien sûr, une telle approche n'est pas limitée à l'utilisation d'un seul serveur, comme le montre la figure 200 où il existe deux serveurs de partage *Serveur de Partage 1* et *Serveur de Partage 2* coopérant l'un avec l'autre.

Ce mécanisme de réplication peut être scindé en deux phases distinctes : la réplication sur un serveur d'une transition effectuée par un objet chez un client (qui en a le droit au sens de l'exclusion mutuelle), et la propagation de cette même transition du serveur chez tous les clients qui possèdent réellement (au sens du mécanisme de chargement progressif) l'objet.

Réplication sur un serveur depuis un poste client

Supposons qu'un objet permanent partagé SPO (pour « Shared Persistent Object » en terminologie anglosaxonne), chez un client C, effectue une transition due à la réception d'un message d'un objet transitoire ou d'un objet permanent privé. Les informations suivantes sont alors envoyées sur le serveur :

1. L'objet partagé qui effectue la transition
2. L'état précédent la transition (que nous appellerons *état précédent*)
3. L'état courant après la transition (que nous appellerons état *courant*)
4. Le temps logique correspondant à cette transition chez le client qui en est à l'origine

5. Le temps logique correspondant à la transition précédente chez le client qui en est à l'origine

Grâce aux informations 1. et 3., l'application sur le serveur retrouve de quel objet il s'agit et lui fait effectuer sa transition dans l'état spécifié. Les informations 2., 4. et 5. servent quant à elles à divers traitements de récupération d'erreurs explicités plus loin dans la section 5 du chapitre VI.

Lorsque l'objet permanent partagé effectue sa transition sur le serveur, il le fait pour un temps logique du Séquenceur de l'application serveur.

Considérons l'exemple où l'on dispose de quatre objets A, B, C et D. A est un objet transitoire, B et C sont des objets permanents partagés et D est un objet permanent privé.

Un extrait des diagrammes d'état-transition de ces objets est illustré sur la figure 201.

Supposons qu'un client K provoque la transition de l'état S0 à S1 de l'objet A. On alors le déroulement illustré sur la figure 202 (les objets ont été mis sous forme canonique conformément à la description faite dans le chapitre IV).

Ceci se traduit au niveau du client K et du serveur de la façon suivante, illustrée sur la figure 203 :

- l 'objet B (partagé) effectue une transition due à un objet transitoire. Cette transition est répliquée sur le serveur.
- l'objet C (partagé) effectue une transition due à un objet permanent privé. Cette transition est répliquée sur le serveur.
- l'objet D est un objet permanent privé, l'application du serveur n'en a pas la visibilité. Le message envoyé par B dans l'état S5B est ignoré.

- puisqu'une application tourne sur le serveur, les transitions des objets partagés se font directement sur le serveur, indépendamment de celles qui ont lieu chez le client K.

Propagation d'une transition répliquée vers les clients concernés

Chaque fois qu'une transition d'un objet permanent partagé a lieu sur le serveur, elle est propagée sur chacun des postes clients qui possède réellement l'objet. Les clients qui ne possèdent un tel objet que virtuellement le récupèreront de toute façon à son état courant (sur le serveur) quand il sera réalisé à nouveau.

Ceci peut être matérialisé par le schéma de la figure 205, à considérer en relation avec le schéma de la figure 204qui donne la représentation d'un objet transitoire T et de trois objets permanents partagés P1, P2 et P3.

Tout d'abord, l'objet T fait transiter l'objet P1, et ceci est repliqué sur le serveur.

Le client L possède réellement les objets P1, P2 et P3. Les transitions de chacun de ces objets sont répliquées depuis le serveur.

Le client M ne possède réellement quant à lui que les objets P2 et P3. Leurs transitions sont répliquées. L'objet P1 n'est là que virtuellement, et sa transition n'est pas répliquée.

Seul l'objet P3 est présent chez le client N, et sa transition est répliquée.

Les transitions qui se situent en aval de l'objet P3 ont lieu indépendamment chez le client K et le serveur.

Réplication dans un cadre multi-serveurs

Il se peut que les objets partagés par des clients soient répartis sur plusieurs serveurs. Le mécanisme de réplication et de propagation présenté dans les chapitres IV et V reste le même pour chaque serveur pris séparément. Chaque serveur exécute l'application et régule les objets qu'il possède au moyen d'un séquenceur.

De la même manière qu'une transition d'un objet partagé provoquée par un objet privé ou transitoire est répliquée par valeur, la transition d'un objet partagé d'un serveur provoquée par un objet partagé d'un autre serveur est également répliquée par valeur. Si d'autres objets partagés de ce serveur sont concernés par le cycle qui en découle, ils sont bien sûr répliqués par calcul. Le schéma de la figure 206 illustre ce mécanisme, dans lequel

- T fait transiter P1, et ceci est repliqué sur le serveur S1, et
- P2 fait transiter P3, et ceci est traité comme si P2 était transitoire ou privé : la réplication se fait par valeur.

Section 4 - Protocole de communication

Dans le cadre de l'exécution d'applications utilisant des objets permanents partagés, un système de communication est requis pour véhiculer des informations du serveur vers plusieurs clients, ainsi que d'un client vers des serveurs.

Pour ce faire, chaque poste client possède un composant qui lui permet de dialoguer avec les différents serveurs. Ces derniers doivent toujours être à l'écoute des éventuelles demandes de connexion pouvant venir des clients.

A chaque connexion, un composant doit être généré sur le serveur. Il est spécifique au client qui vient de se connecter, se charge d'interpréter les informations en provenance du client, et d'envoyer à celui-ci les informations du serveur.

- Composant sur les postes clients

Le composant réseau du poste client (clientCommunicationManager) offre les fonctionnalités suivantes :

- ✓ Etablir une connexion avec le serveur, c'est à dire initier la création du composant sur le serveur qui lui correspond, et instancier un nouvel objet GbxClient, propre au client qui se connecte.
- ✓ Ecouter les informations qui viennent du serveur. Ces informations ne pouvant être que des appels de méthodes.
- ✓ Envoyer des informations au serveur, à l'aide d'une méthode du type
  clientCommunicationManager.send(methodName,argList)
  Où methodName est le nom de la méthode à appeler.
  argList est la liste des arguments de cette méthode.

- Composant sur les serveurs

De manière symétrique, le composant réseau du serveur pour un poste client doit permettre :

- ✓ D'écouter les informations qui viennent du poste client auquel il est « relié », ces informations étant du type (methodName,argList) comme nous l'avons vu. methodName doit correspondre à une méthode de l'objet GbxClient du serveur.
- ✓ D'envoyer des informations au client, à l'aide du même type de méthode serverCommunicationManager.send(methodName,argList)

Etablissement d'une connexion

En référence maintenant à la figure 207, on observe que chaque serveur possède un composant Listener qui écoute les éventuelles demandes de connexion des postes clients. Lorsqu'une de ces demandes arrive, il provoque la création sur le serveur d'un composant réseau (ServerCommunicationManager) qui va dialoguer avec son correspondant sur le poste client (ici Client1 ou Client2). Il est constitué d'un routine qui écoute les différents messages pouvant provenir du poste client distant

correspondant, et d'un dispositif (appel de méthode) lui permettant d'envoyer des données à celui-ci.

A la connexion d'un client, le composant ServerCommunicationManager appelle la méthode newClient(connexionRef) de l'objet Document du serveur. Cette méthode crée une instance de l'objet GbxClient, qui va être le représentant du poste client au sein du serveur. Au moment cette instance est créée, l'objet Document passe une référence sur lui-même à GbxClient. La référence de ce GbxClient va être retournée au composant réseau du serveur qui a commandé son instanciation.

Suite à cela, le composant réseau du serveur renvoie au composant réseau du client qui s'était connecté un accusé de réception lui indiquant que la connexion s'est bien effectuée. Cet accusé de réception sera utilisé pour éviter la commutation automatique vers un système de communication de secours utilisant le protocole http. Le processus de connexion est présenté dans le diagramme de la figure 208.

Communication entre clients et serveurs

En référence maintenant à la figure 209, les seules informations qui peuvent être échangées entre clients et serveurs sont des noms de méthodes, et des arguments pour ces méthodes. Comme nous l'avons vu précédemment, ces informations sont passées à l'aide de la méthode send des composants réseaux.

serverCommunicationManager.send(methodName,argList)

clientCommunicationManager.send(methodName,argList)

Si le destinataire est le serveur, la méthode methodName est appelée chez l'objet GbxClient, et si le destinataire est le poste client, la méthode methodName est appelée chez l'objet Application du poste client. Un procédé de contrôle de cohérence peut être mis en place après la transmission pour vérifier que les arguments figurant dans argList aient un sens.

Les diagrammes de séquence des figures 210 et 211 illustrent les deux types de communication, respectivement d'un client vers le serveur, et du serveur vers un client.

Fin d'une connexion

Lorsqu'un utilisateur quitte toutes les applications utilisant des objets permanents partagés, la connexion peut être coupée. Pour ce faire, l'objet Document sur le poste client appelle la méthode endOfConnexion(argument) de son composant réseau.

Ce composant doit alors signaler au composant réseau qui lui correspond sur le serveur qu'il doit supprimer l'instance de GbxClient, et disparaître. Le composant réseau sur le serveur appelle pour cela la méthode killClient(clientRef) de l'objet Application sur le serveur.

Ceci peut être représenté par le diagramme de séquence illustré sur la figure 212.

Gestion des problèmes d'accès sécurisés

Dans le cas de restrictions d'accès (« firewalls », terminologie anglo-saxonne pour murs anti-feu) pour des utilisateurs n'ayant pas les pleins pouvoirs sur la gestion de leur domaine, une migration automatique vers un système de secours est envisagée. Un tel système présente les caractéristiques suivantes :

- Commutation automatique

  Par défaut, le protocole de transmission utilisé est celui décrit précédemment. Il faut cependant y inclure un système de gestion d'erreurs qui passe automatiquement au système décrit ci-dessous en cas de problème d'émission ou de réception.

Lors de la demande de connexion, le composant réseau du client qui se connecte démarre le décompte d'une temporisation. Si cette temporisation arrive à expiration avant que le composant réseau client ait reçu l'accusé de réception, le protocole commute.

- Communication d'un client vers le serveur

  Pour contourner les restrictions d'accès, l'idée est d'utiliser le protocole « http » pour véhiculer les informations. Pour ce faire, les éléments methodName et argList qui doivent être transmis d'un client vers un serveur sont passés en paramètres lors de l'appel d'un fichier de type librairie de liens dynamiques DLL (pour le terme anglo-saxon « Dynamic Link Library ») du serveur, via une adresse de type URL. Le module CommunicationManager du poste client sera chargé d'un tel appel.

  A la réception, le module CommunicationManager du module GbxClient associé sur le serveur au client concerné devra récupérer ces paramètres et procéder à l'appel de la méthode methodName(argsList) du module GbxClient du serveur.

- Communication d'un serveur vers un client

  L'idée est la même que pour le cas précédent, sauf que dans le cadre du protocole http, il est impossible au serveur d'envoyer par lui-même des données au client. Pour contourner cette difficulté, on utilise un procédé de pseudo-poussée (« push » d'informations par exemple du type suivant : le module CommunicationManager du poste client, à intervalles de temps réguliers (par exemple toutes les 10 secondes...), effectue une requête http sur le serveur pour récupérer un fichier au format html. Ce fichier, géré par le module CommunicationManager correspondant sur le serveur, comporte les données que le module GbxClient attend. Le module CommunicationManager du poste client doit donc être à même d'extraire les éléments methodName et argsList de ce fichier html, et d'appeler la méthode correspondante methodName(argsList) de l'objet Application du poste client.

Section 5 - Exclusion mutuelle

Du fait que les mêmes instances sont partagées par plusieurs utilisateurs, un mécanisme d'exclusion mutuelle est requis pour éviter les accès concurrents à un même objet. Ce mécanisme est basé sur un système à double verrous : un verrou en lecture/écriture (W/R) et un verrou en lecture (R).

Toute modification de l'état d'un objet permanent partagé ne peut être faite que si l'objet est déverrouillé en lecture/écriture. Par contre, l'accès en lecture aux valeurs caractéristiques de l'objet (à son état par exemple) ne nécessite que le déverrouillage R.

La récupération du verrou est à la charge de l'objet qui en fait sentir le besoin, c'est à dire qui modifie l'état de l'objet permanent partagé (il demande alors le verrou W/R) ou qui lit l'état courant de l'objet (il demande alors un verrou R).

L'attribution du verrou W/R a un caractère exclusif. Seul un client peut accéder à un objet en lecture/écriture à un instant donné. Mais ce n'est pas le cas pour l'accès en lecture : ainsi plusieurs utilisateurs peuvent accéder en lecture au même objet. Bien entendu, le fait qu'un client possède le verrou W/R exclut pendant le même temps toute obtention d'un verrou R par un quelconque autre client.

Utilisation des objets verrouillés

Un client peut utiliser un objet permanent partagé à partir du moment où il obtient un verrou (que ce soit R ou W/R) sur cet objet. Tant que le cycle (au sens où il a été défini plus haut dans le chapitre IV), initié par la transition de l'objet qui a provoqué la demande de verrou, n'est pas terminé, l'objet verrouillé est considéré comme étant en cours d'utilisation.

On considérera donc qu'un client a fini d'utiliser un objet verrouillé en lecture si le cycle au cours duquel il lit une des valeurs de cet objet est terminé.

Par exemple, et en référence à la figure 213,supposons qu'un objet A veuille envoyer un message à un objet B en lui passant l'état courant d'un objet O permanent partagé. On considère que l'objet A a fini d'utiliser l'objet O lorsque l'objet A reprend la main par retour d'appel de méthode.

On considère également qu'un client a fini d'utiliser un objet déverrouillé en lecture/écriture si le cycle provoqué par la transition de l'objet qui a demandé le verrou est terminé.

Par exemple, et en référence à la figure 214, supposons qu'un objet A obtient le verrou WR sur un objet O permanent partagé pour le faire changer d'état. On considère que l'objet A a fini d'utiliser l'objet O lorsque l'objet A reprend la main par retour d'appel de méthode.

Comportement du verrou d'un objet

Un verrou R ou W/R d'un objet va être défini ci-dessous à l'aide de diagrammes d'état-transition.

Pour un objet partagé O donné, un client peut être dans un des états suivants vis à vis du verrou R ou du verrou W/R :

- Le client peut ne pas posséder le verrou sur l'objet O
- Le client peut posséder le verrou sur l'objet O mais ne pas l'utiliser
- Le client peut posséder le verrou sur l'objet O et l'utiliser
- Le client peut être en passe de rendre le verrou sur l'objet O si un autre client en a fait la demande.

Un objet permanent partagé peut être réalisé de deux façons distinctes chez un client (juste au moment où le client a besoin de l'objet ou par anticipation, voir l'introduction du chapitre V), ce qui influe sur l'attribution du verrou.

En effet, si un objet permanent partagé est réalisé selon un mode « juste à temps », cela signifie que le client veut lire ou manipuler cet objet. Le verrou R ou W/R lui est alors donné directement. On a alors le diagramme d'état-transition de la figure 215 :

En revanche, si un objet permanent partagé est réalisé par anticipation, le client n'a pas encore émis la volonté de manipuler l'objet. Il n'obtiendra le verrou que s'il accède véritablement à l'objet. Ceci se manifeste dans le diagramme d'état-transition de la figure 216.

Demande de verrou WR

Un objet A chez un client C demande le verrou sur l'objet auquel il désire envoyer un message (s'il ne l'a pas déjà).

Si l'objet n'est pas déjà utilisé par un autre client (que ce soit en lecture ou en écriture), il le devient comme le montre le diagramme de séquence illustré sur la figure 217.

Si l'objet était déverrouillé en lecture, le verrou est ôté à chaque client qui le possède au terme d'un certain temps de latence. Ce temps de latence est propre à chaque objet et peut en constituer par exemple un attribut.

Ceci est illustré sur la figure 218.

Si l'objet était déjà verrouillé en lecture/écriture, le verrou est rendu également après un certain temps de latence. Ce cas peut être représenté par le schéma de la figure 219.

Comme on se trouve ici dans le cadre d'un protocole de communication basé sur le protocole TCP, et que ce protocole assure une communication selon le mode « premier entré- premier sorti » (FIFO), toutes les transitions de l'objet B répliquées depuis le Client 1 ont eu lieu chez le Client 2 avant que le verrou sur B n'ait été obtenu.

Demande de verrou R

L'hypothèse est qu'un objet O chez un client C demande le verrou R sur l'objet dont il veut connaître l'état (s'il ne le connaît pas déjà).

Si l'objet n'est verrouillé ni en lecture/écriture ni en lecture, il devient verrouillé en lecture, comme illustré sur la figure 220.

Si l'objet est déjà utilisé en lecture par d'autres clients, l'objet C est ajouté à la liste des clients qui possèdent le verrou R sur l'objet O. Plusieurs clients accèdent ainsi au même objet en lecture. Si l'objet est déjà utilisé en lecture/écriture par un autre

client, la requête de l'objet O est placée en attente jusqu'à ce que l'objet soit disponible. ceci est illustré sur la figure 221.

Gestion des blocages mutuels

Il demeure cependant des cas dits de blocage mutuel où des clients peuvent s'attendre les uns les autres à cause de demande de verrous.

Considérons par exemple, en référence aux figures 222 et 223, deux clients C1 et C2. Le client C1 possède les objets permanents partagés O1 et O2, tandis que le client C2 possède les objets permanents partagés O1, O2 et O3. Les clients possèdent des verrous comme suit :

le client C1 possède le verrou W/R sur O1.

le client C2 possède le verrou W/R sur O2 et O3.

L'on se trouve alors dans la situation de la figure 224.

Supposons maintenant que, au cours du cycle initié par l'objet O1, le client C1 demande au client C2 le verrou W/R sur l'objet O2. Le client C2 ne peut pas rendre le verrou sur l'objet O2 car un cycle initié par l'objet O2 est en cours chez le client C2. L'on se trouve alors dans la situation de la figure 225.

Le client C1 attend que le client C2 ait fini d'utiliser l'objet O2 pour prendre le verrou et continuer son exécution. Supposons maintenant que dans le cycle initié par l'objet O2 chez le client C2, ce même client en vient à demander le verrou sur l'objet O1. Comme nous l'avons vu, un cycle initié par l'objet O1 est en cours chez le client C1, et le verrou W/R sur l'objet O1 ne peut donc pas être rendu. Ceci est illustré sur la figure 197 : le client C1 attend après le client C2 pour obtenir le verrou sur l'objet O2, tandis que le client C2 attend après le client C1 pour obtenir le verrou sur l'objet O1 ; l'on se trouve dans une situation de blocage mutuel.

Pour lever cette situation, il faut défaire les actions effectuées par l'un des deux clients de façon à ce que tous les verrous qui interviennent dans la boucle de blocage mutuel soient restitués.

Supposons que dans le même exemple, les obtentions de verrou ont eu lieu dans l'ordre illustré sur la figure 225. Il semble ici plus pertinent de défaire les actions qui se sont déroulées chez le client C2. En effet, la première obtention d'un verrou sur un objet figurant dans la boucle de blocage mutuel est plus récente chez le client C2.

Défaire les actions effectuées chez le client C2 revient à utiliser une fonctionnalité dite de « Retour arrière » (« Rollback » en terminologie anglo-saxonne) décrite ci-après à partir du temps logique où le client C2 a obtenu le verrou sur l'objet O2. L'objet O1 peut ainsi obtenir le verrou sur l'objet O2 et continuer son cycle. Le client C2 est alors mis en attente jusqu'à ce que le client C1 libère l'objet O2.

Cependant, le fait de résoudre le blocage mutuel implique de détecter la présence d'une boucle dans les demandes de verrou, comme on va maintenant le décrire.

Détection des boucles

Il ne peut y avoir une boucle dans les demandes de verrou si un objet reçoit une demande de verrou alors que lui-même est en attente d'un verrou. Cependant, ce n'est pas parce qu'on est dans ce cas qu'il y a forcément une boucle.

C'est en tout cas à partir de ce cas litigieux que la détection de boucle commence.

La figure 226 illustre ainsi dans sa partie gauche un cas de blocage mutuel, tandis que sa partie droite illustre un cas où il n'y a pas de blocage mutuel.

Pour éviter de mélanger les données liées à chaque instance de l'algorithme de gestion des blocages mutuels, ces données sont estampillées avec des marqueurs à

l'aide d'un compteur croissant, représentatif par exemple de l'instant de lancement de l'instance de l'alogrithme.

Ainsi, et maintenant en référence à la figure 227, lorsque l'objet A reçoit une demande de verrou alors qu'il en déjà en attente, il relève le temps CPU de la séquence en cours afin de s'en servir comme marqueur.

Il remonte ensuite les demandes de verrou de cette séquence en marquant chaque objet en attente par le temps CPU qu'il a relevé. S'il retombe sur lui-même pour la même marque, un cycle est détecté, et le processus de retour en arrière peut être mis en œuvre.

Section 6 - Mécanisme de retour en arrière (RollBack)

Deux approches d'exécution répliquées de ce mécanisme sont possibles : pessimiste et optimiste. Dans l'approche optimiste que nous décrivons dans le présent chapitre, le mécanisme de retour en arrière sert de procédé de rattrapage de prédictions erronées. Le même mécanisme de Rollback est aussi utilisé dans la gestion des blocages mutuels dans le cadre de l'exclusion mutuelle. Ci-après, nous traitons le cas du rattrapage, sur un exemple.

Dans cet exemple, illustré sur la figure 228, les objets O1, O2, O3 et O4 sont des objets partagés, tandis que l'objet O5 est un objet privé, et l'on suppose que le client Client1 possède le verrou W/R sur les objets partagés.

Suite à une action d'un utilisateur sur l'objet O1, les messages Msg1, Msg2 et Msg3 s'enchaînent. Comme elle concerne des objets partagés, la transition de l'objet O1 est répliquée par valeur sur un serveur. Par contre, les conséquences de la transition de l'objet O1, à savoir les transitions de O2 et de O3 ne sont pas transmises puisque le serveur les génère lui-même.

Une fois la transition de l'objet O3 effectuée (suite au message Msg3), le premier cycle issu de l'action de l'utilisateur est terminé. Les traces dont disposent les séquenceurs du poste client de l'utilisateur et du serveur contiennent alors respectivement les informations illustrées sur la figure 229.

Sur le serveur, une fois le premier cycle terminé, le séquenceur entre en activité et traite les actions PutNext figurant dans la trace. Ceci se traduit par l'envoi du message Next aux objets qui en avaient fait la demande (ici l'objet O2) et du message Msg5 qui s'ensuit, comme illustré sur la figure 230.

La trace sur le serveur contient alors les informations illustrées sur la figure 231.

Supposons maintenant que sur son poste, l'utilisateur du client client1 manipule l'objet O5 pendant le premier cycle (qui était initié par le message Msg1). Il se déroule alors chez le client le scénario illustré sur la figure 232.

Le séquenceur du client1 contient alors les informations illustrées sur la figure 233.

On observe ici que le serveur a pris trop d'avance dans le traitement de son séquenceur, si bien que des scénarios différents sont joués chez le client client1 et sur le serveur. Le serveur doit donc revenir en arrière sur l'avance qu'il a prise : c'est le mécanisme de retour en arrière, dont on va maintenant décrire le fonctionnement.

Détection

Un tel cas litigieux est détecté lors de la réplication de la transition de l'objet O2 de l'état S1 à l'état S4, dont la cause est due à un objet non partagé. En effet, lors d'une réplication, le serveur reçoit les informations suivantes :

1. L'objet qui effectue la transition
2. L'état précédent la transition (que l'on appellera *état précédent*)

3. L'état courant après la transition (que l'on appellera état *courant*)
4. Le temps logique correspondant à cette transition chez le client qui en est à l'origine
5. Le temps logique correspondant à la transition précédente chez le client qui en est à l'origine.

Les informations 1 et 3 permettent au serveur de trouver quel objet il doit faire transitionner et dans quel état. L'information 4 va permettre la détection d'un cas litigieux (si le serveur a déjà associé ce temps logique à un put, il y a conflit). Enfin l'information 5 va servir à repositionner le curseur du traitement dans le cas où une opération de retour en arrière est nécessaire.

Dans le même exemple, le serveur va savoir que la transition de l'objet O2 vers l'état S4 correspond au temps logique 127 pour le client client1. Or le serveur a déjà, par calcul, associé au temps logique 127 une transition qui est différente, puisqu'elle concerne le passage de l'objet O4 vers l'état S1). Un retour en arrière doit donc être effectué.

Etapes du retour en arrière

Le retour en arrière s'effectue par la succession d'étapes suivante :

1. Trouver dans la trace la dernière position « stable » avant le cas litigieux (point à partir duquel le séquenceur va reprendre son traitement)
2. Déterminer dans l'historique quelles sont les entrées qui doivent être défaites
3. Déterminer quels sont les objets touchés et dans quel état ils doivent être restaurés, et les restaurer sur le serveur et chez tous les clients passifs qui les possèdent réellement

4. Répliquer la transition qui a permis de détecter le conflit (ici le passage de l'objet O2 de l'état S1 à l'état S4)

5. Reprendre le traitement du séquenceur à partir de la dernière position stable.

Détail des étapes

- Etape 1

Une situation de conflit peut être schématisé par le diagramme d'état-transition de la figure 234.

La solution au conflit consiste à ramener l'objet qui pose problème au dernier de ses états pour lequel à la fois le client et serveur sont en accord. Plutôt que de rechercher cet état dans la trace du client, l'objet peut lui-même renseigner quel est le temps logique auquel il était entré dans l'état précédent. Ce temps logique est celui du dernier état pour lequel client et serveur sont en accord pour l'objet qui pose problème : ce temps logique est alors transmis au serveur pour que celui-ci puisse s'y ramener.

A chaque fois qu'ils effectuent une transition, les objets stockent pour quel temps logique du client ils changent d'état. Ainsi, lorsqu'une transition est répliquée, le serveur peut recevoir le temps logique auquel l'objet était entré dans l'état précédent.

Donc, lors de la réplication de la transition au temps logique 127, le serveur reçoit le temps logique du client client1 correspondant à la précédente transition de l'objet O2 : ce temps logique est 124.

Le serveur repositionne donc le curseur du séquenceur au temps logique 124 du client client1, comme illustré sur la figure 235.

- Etape 2

Chercher dans la trace du serveur ce qui doit être défait revient à déterminer quelle a été la première transition à avoir été effectuée à tort. C'est à partir de cette transition que ce qui a été effectué en avance sur le serveur doit être défait.

Cette transition est la première transition de l'objet qui pose problème (ici l'objet O2) située en aval (au sens du temps logique) du curseur de traitement (tel qu'il a été replacé à l'étape a).

Dans l'exemple, il s'agit de la transition au temps logique126. C'est donc là qu'est repositionné le curseur d'écriture. Cette transition correspond à la réaction de l'objet O2 à un message Next du séquenceur. Si le serveur avait été en adéquation avec le client client1, ce Next aurait dû être obsolète.

Comme nous l'avons vu, l'objet O2 se trouve encore dans l'état S1 lorsqu'il reçoit le message Next, et celui-ci provoque donc une transition.

La situation correspondante est illustrée sur la figure 236.

Comme on va le voir, seules les transitions figurant à partir du curseur d'écriture sur la figure 236 doivent être annulées, alors que le traitement reprend à partir de la nouvelle position du curseur de traitement. Ceci est justifié par le fait que les conséquences des transitions figurant entre le curseur de traitement et le curseur d'écriture se trouvent forcément après la nouvelle position du curseur d'écriture : elles sont donc défaites. C'est pourquoi les transitions qui en sont à l'origine (donc celles qui figurent à partir du curseur de traitement jusqu'au curseur d'écriture exclu) doivent être conservées pour poursuivre le processus en lieu et place de ce qui a été défait.

- Etape 3

A partir du curseur d'écriture, il faut parcourir la trace pour déterminer quels sont les objets qui ont effectué à tort des transitions.

La trace porte, pour chaque entrée, quel objet a effectué une transition et quel était son état précédent. Il suffit donc de récupérer la première entrée à partir du curseur d'écriture pour chaque objet : on y trouve l'état précédent dans lequel l'objet doit être restauré.

Dans l'exemple, seul l'objet O4 possède une entrée sous le curseur d'écriture : il doit être restauré dans l'état S0.

Les transitions effectuées en avance et à tort par le serveur doivent être annulées. Elles doivent également être annulées chez tous les clients qui les ont passivement répliquées et qui les possèdent réellement ces objets (au sens du mécanisme de chargement progressif décrit plus haut).

Pour ce faire, tous les objets devant l'être sont restaurés dans leur état précédent. Ces transitions forcées sont au passage répliquées chez les clients qui possèdent réellement ces objets en utilisant le procédé de réplication déjà décrit (Etape 4.).

Une fois ceci réalisé, toutes les entrées de l'historique qui figurent en aval (au sens du temps logique) du curseur d'écriture sont supprimées, comme illustré sur la figure 237.

- Etape 4

Une fois que les transitions effectuées à tort ont été défaites, il faut réhomogénéiser l'ensemble du système. Pour ce faire, on fait en sorte que la transition répliquée qui a permis de détecter le cas litigieux ait aussi lieu sur le serveur. Le mécanisme de

séquencement fait alors que les informations correspondant à cette transition soient reportées dans la trace à partir du curseur de trace. Ceci est illustré sur la figure 238.

Bien sûr, cette transition est propagée chez tous les clients qui possèdent réellement l'objet (au sens du mécanisme de chargement progressif).

- Etape 5

Une fois la transition répliquée, le séquenceur peut reprendre la main pour gérer les activités en cours des objets. Il reprend son traitement à partir du curseur de traitement.

Dans l'exemple, sa première tâche sera d'envoyer le message Next à l'objet O2. Cependant, l'objet O2 se trouvant maintenant dans l'état S4, le Next qu'il reçoit est obsolète et est donc ignoré.

Bien entendu, la présente invention n'est nullement limitée aux formes de réalisations décrites et représentées, mais l'homme du métier saura y apporter toute variante ou modification conforme à son esprit. En particulier, les caractéristiques de l'invention énoncées dans les différents chapitres et les différentes sections de ces chapitres peuvent être combinées de manières très diverses.

ANNEXE

Composant BEHAVIOR

```
<PUBLIC:ATTACH EVENT="ondragstart"
ONEVENT="InitiateDrag"/>
<PUBLIC:ATTACH EVENT="ondrop"       ONEVENT="FinishDrag"/>
<PUBLIC:ATTACH EVENT="ondragenter" ONEVENT="DragEnter"/>
<PUBLIC:ATTACH EVENT="ondragleave" ONEVENT="DragLeave"/>

<SCRIPT LANGUAGE="JScript">
// ------------------------------------------------------
-----------------
// gestion du drag'n drop

// ->L'utilisateur commence à déplacer la poignée...
// Met en mémoire l'identifiant du Container source.
function InitiateDrag()
{
     var caid = event.srcElement.containerid ;
     var ceid = event.srcElement.contentid ;
     event.dataTransfer.setData("Text","#contentid="+ceid
+"#containerid="+caid) ;
     event.dataTransfer.effectAllowed = "copy" ;
}

// ->L'utilisateur place la poignée sur une autre...
// Modifie le style pour montrer l'interaction de la
cible.
function DragEnter()
{
     event.dataTransfer.dropEffect='copy' ;
     style.backgroundColor='red' ;
     style.color='white' ;
     event.returnValue=false ;
}

// ->L'utilisateur éloigne la poignée de l'autre...
// Re-modifie le style pour remettre la cible à un état
normal.
function DragLeave()
{
     style.backgroundColor='' ;
     style.color='' ;
}
```

```
    // ->L'utilisateur lache la poignée sur l'autre…
    // Récupère l'identifiant du Container source et de la
    cible.
    // Transmet l'action par événement au système .
    function FinishDrag()
    {
         style.backgroundColor='' ;
         style.color='' ;
         // Diffusion de l'évènement "onderive"...
         var oEvent     = createEventObject() ;
         oEvent.cible   = event.srcElement.containerid ;
         oEvent.source  =  event.dataTransfer.getData("Text")
    ;
         fire("onderive",oEvent) ;
    }
    // Fin de gestion du drag'n drop
    // ----------------------------------------------------
    -----------------
    </SCRIPT>
```

REVENDICATIONS

1. Procédé de gestion de l'affichage, sur l'écran d'affichage d'un poste informatique, de fenêtres d'un système d'exploitation lors d'une opération de glisser-déposer d'un objet contenu dans une première fenêtre (F1) vers une seconde fenêtre (F2), les première et seconde fenêtres étant affichées de telle sorte que la première fenêtre recouvre partiellement la seconde fenêtre, caractérisé en ce qu'il comprend les étapes successives suivantes :

(a) détection des mouvements d'un pointeur actionné par l'utilisateur au cours d'une opération de glisser d'un objet préalablement saisi dans la première fenêtre, pour déterminer si ledit pointeur se situe dans la zone de la seconde fenêtre non recouverte par la première fenêtre, et

(b) lorsque ledit pointeur a séjourné dans la zone de la seconde fenêtre non recouverte par la première fenêtre et qu'au moins une autre condition est remplie, affichage de la seconde fenêtre (F2) dans son entier de manière à rendre accessible au pointeur l'ensemble de ladite seconde fenêtre.

2. Procédé selon la revendication 1, caractérisé en ce que ladite autre condition est déterminée par le fait que ledit pointeur atteint un bord de la première fenêtre alors que ledit pointeur se situait dans la zone de la seconde fenêtre non recouverte par la première fenêtre.

3. Procédé selon l'une des revendications 1 et 2, caractérisé en ce que l'étape d'affichage de la seconde fenêtre dans son entier comprend une sous-étape de déplacement de l'affichage de la première fenêtre pour qu'elle ne recouvre plus ladite seconde fenêtre.

4. Procédé selon les revendications 2 et 3 prises en combinaison, caractérisé en ce que le déplacement de l'affichage de la première fenêtre est corrélé aux déplacements du pointeur après qu'il a atteint le bord de la première fenêtre.

5. Procédé selon l'une des revendications 2 à 4, caractérisé en ce qu'il comprend en outre une étape consistant à effectuer une signalisation auprès de l'utilisateur lors de l'atteinte par le pointeur du bord de la première fenêtre.

6. Procédé selon la revendication 5, caractérisé en ce que ladite signalisation est choisie dans le groupe comprenant une signalisation visuelle, une signalisation sonore et une signalisation tactile telle qu'un retour de force au sein d'un périphérique d'entrée manuel qui commande ledit pointeur.

7. Procédé selon l'une des revendications 1 à 6, caractérisé en ce que ladite autre condition est déterminée par le fait que ledit pointeur a séjourné dans la zone de la seconde fenêtre non recouverte par la première fenêtre pendant une durée supérieure à un seuil prédéterminé.

8. Procédé selon la revendication 1, caractérisé en ce que ladite autre condition est déterminée par le fait qu'une touche d'un clavier du poste informatique est actionnée.

9. Procédé selon la revendication 1, caractérisé en ce que ladite autre condition est déterminée par la réception dans le poste informatique d'une commande vocale.

10. Procédé selon l'une des revendications 1 à 9, caractérisé en ce qu'il comprend en outre l'étape additionnelle :

(c) lorsque ledit pointeur a séjourné dans la zone de la seconde fenêtre qui était précédemment recouverte par la première fenêtre et qu'au moins une seconde autre condition est remplie, ré-affichage de la première fenêtre (F1) de manière qu'elle recouvre partiellement la seconde fenêtre comme précédemment.

11. Procédé selon la revendication 10, caractérisé en ce que la première autre condition est le franchissement par le curseur du bord de la première fenêtre et en ce que la seconde autre condition est un déplacement inverse du pointeur.

12. Procédé selon la revendication 1, caractérisé en ce qu'il est mis en œuvre sur un poste informatique client, en ce que les contenus des fenêtres sont aptes à être chargés à partir d'un poste informatique serveur, en ce que le contenu de la première fenêtre est à chargement rapide, en ce que le contenu de la seconde fenêtre est à chargement plus lent, et en ce que:

- le contenu de la première fenêtre est affiché dès après son chargement,

- l'étape de détection est démarrée seulement à la fin du chargement du contenu de la seconde fenêtre.

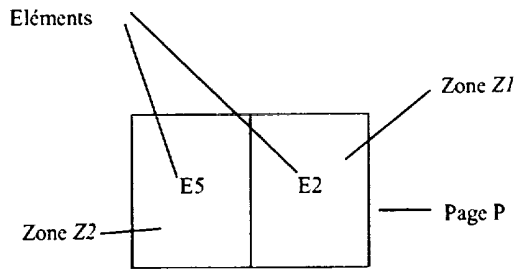


Fig. 1

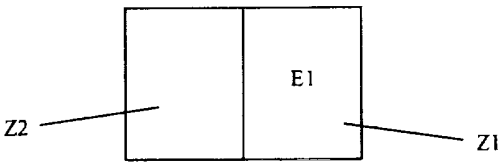


Fig. 2

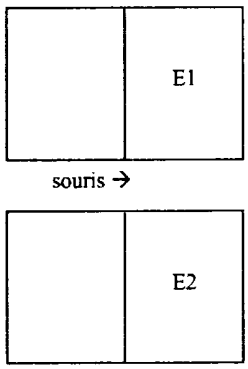


Fig. 3

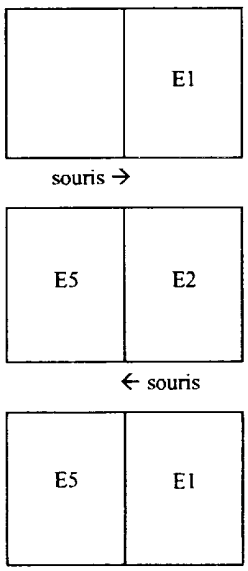


Fig. 4

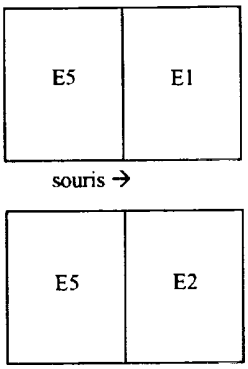

E5
E1
souris →
E5
E2

Fig. 5

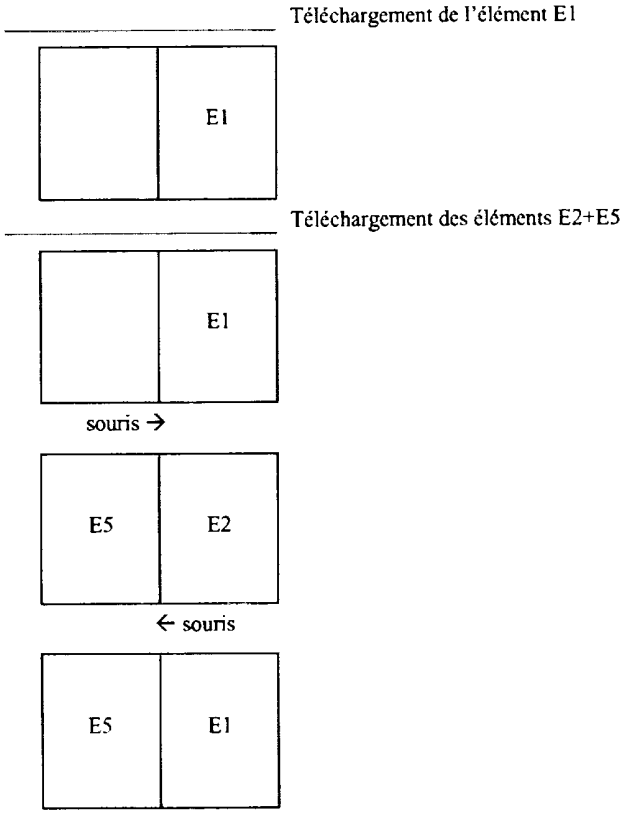

Téléchargement de l'élément E1
E1
Téléchargement des éléments E2+E5
E1
souris →
E5
E2
← souris
E5
E1

Fig. 6

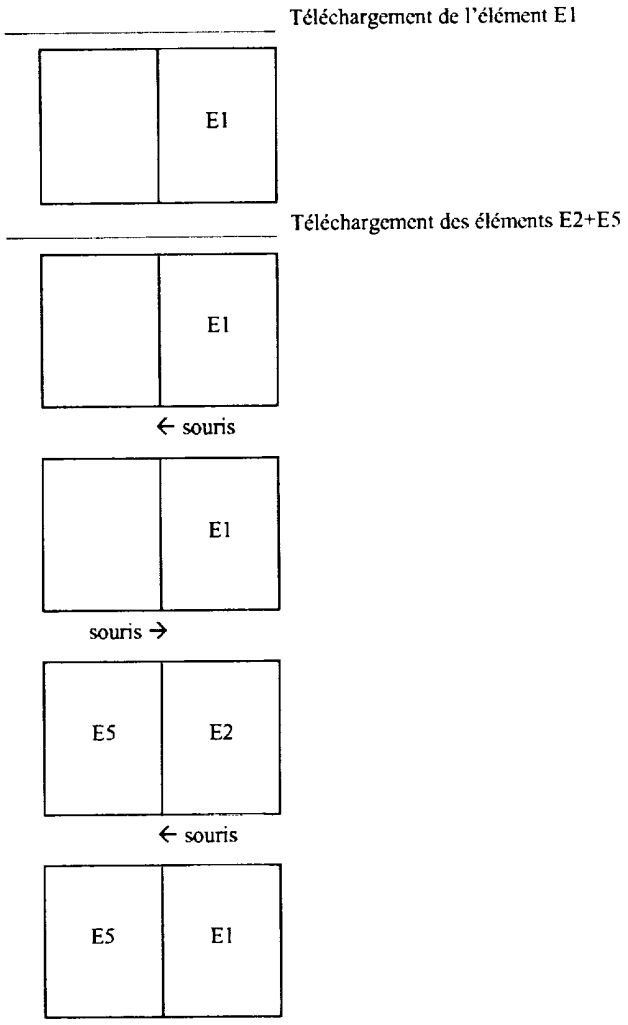

Téléchargement de l'élément E1
E1
Téléchargement des éléments E2+E5
E1
← souris
E1
souris →
E5
E2
← souris
E5
E1

Fig. 7

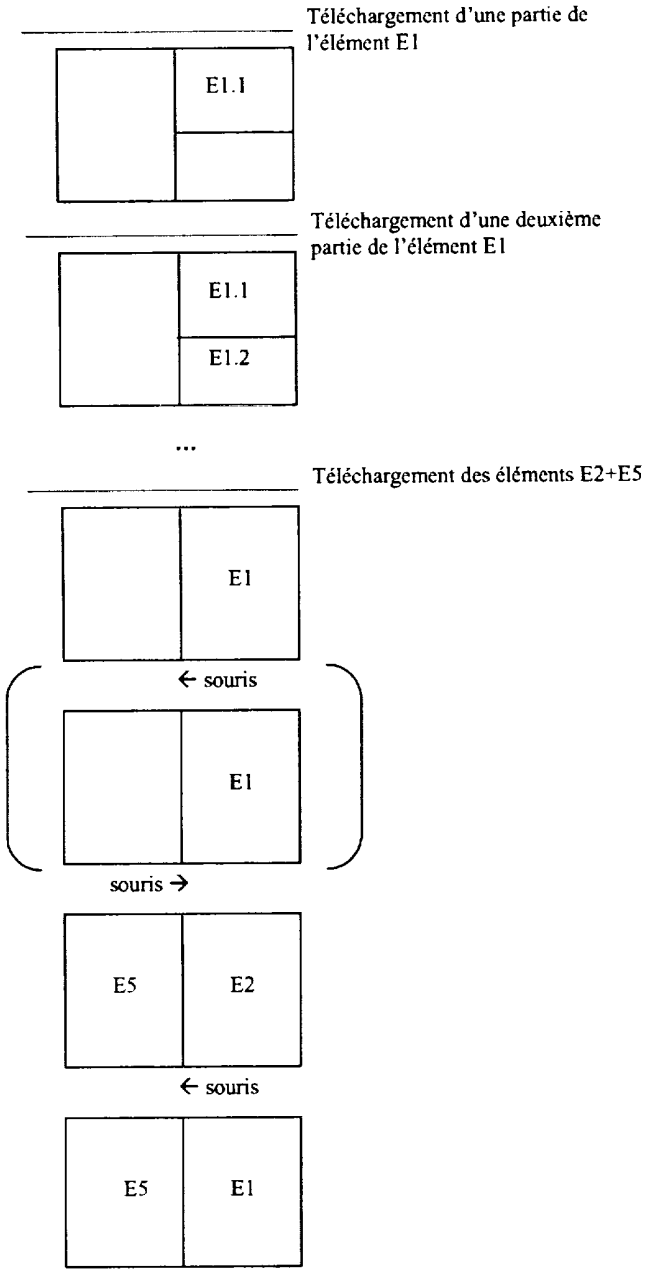

Téléchargement d'une partie de l'élément E1
E1.1
Téléchargement d'une deuxième partie de l'élément E1
E1.1
E1.2
...
Téléchargement des éléments E2+E5
E1
← souris
E1
souris →
E5
E2
← souris
E5
E1

Fig. 8

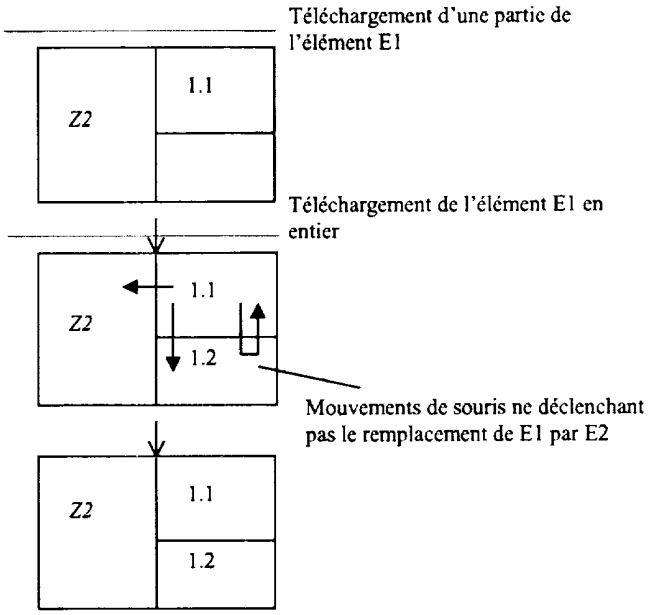

Téléchargement d'une partie de l'élément E1
Z2
1.1
Téléchargement de l'élément E1 en entier
Z2
1.1
1.2
Mouvements de souris ne déclenchant pas le remplacement de E1 par E2
Z2
1.1
1.2

Fig. 9

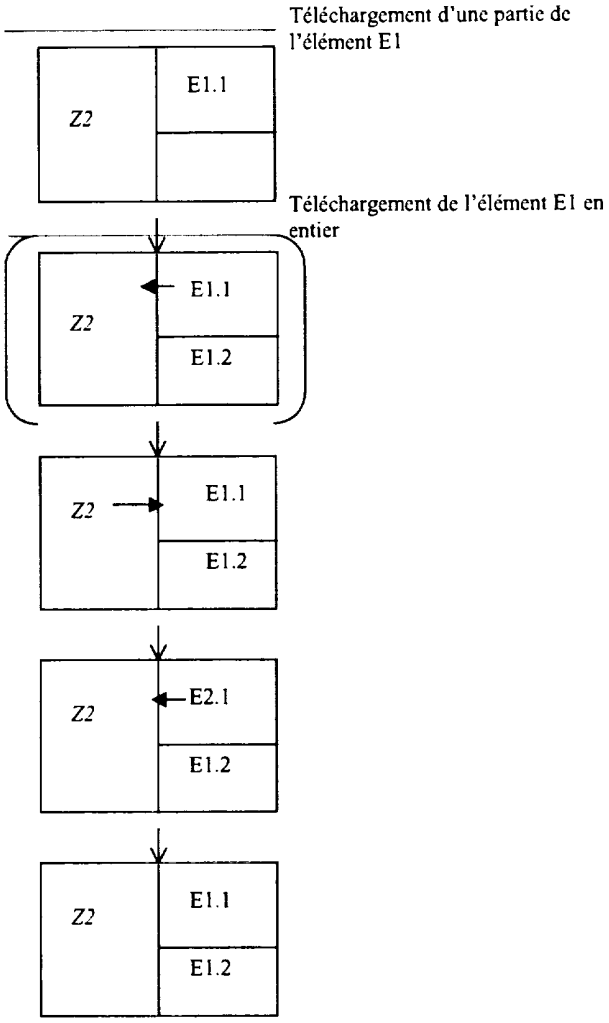

Téléchargement d'une partie de l'élément E1
Z2
E1.1
Téléchargement de l'élément E1 en entier
Z2
E1.1
E1.2
Z2
E1.1
E1.2
Z2
E2.1
E1.2
Z2
E1.1
E1.2

Fig. 10

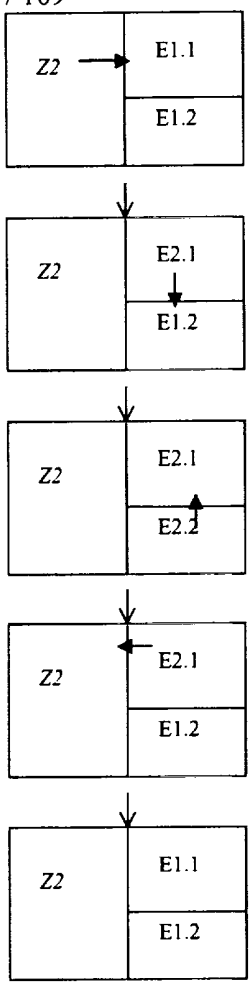


Fig. 11

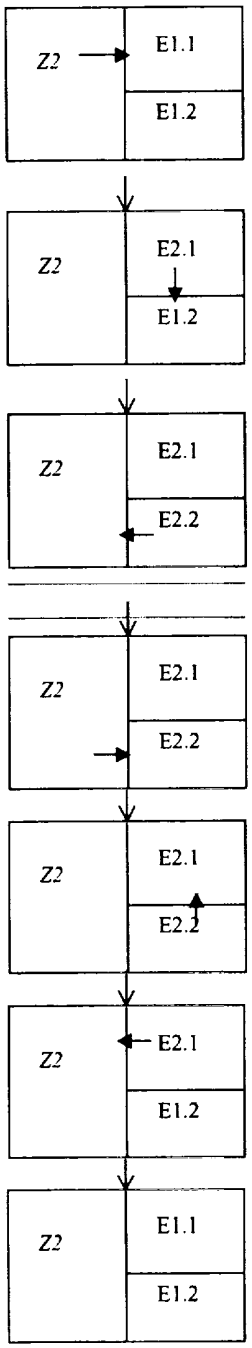

Z2
E1.1
E1.2
Z2
E2.1
E1.2
Z2
E2.1
E2.2
Temps écoulé < seuil de réversibilité
Z2
E2.1
E2.2
Z2
E2.1
E2.2
Z2
E2.1
E1.2
Z2
E1.1
E1.2

Fig. 12

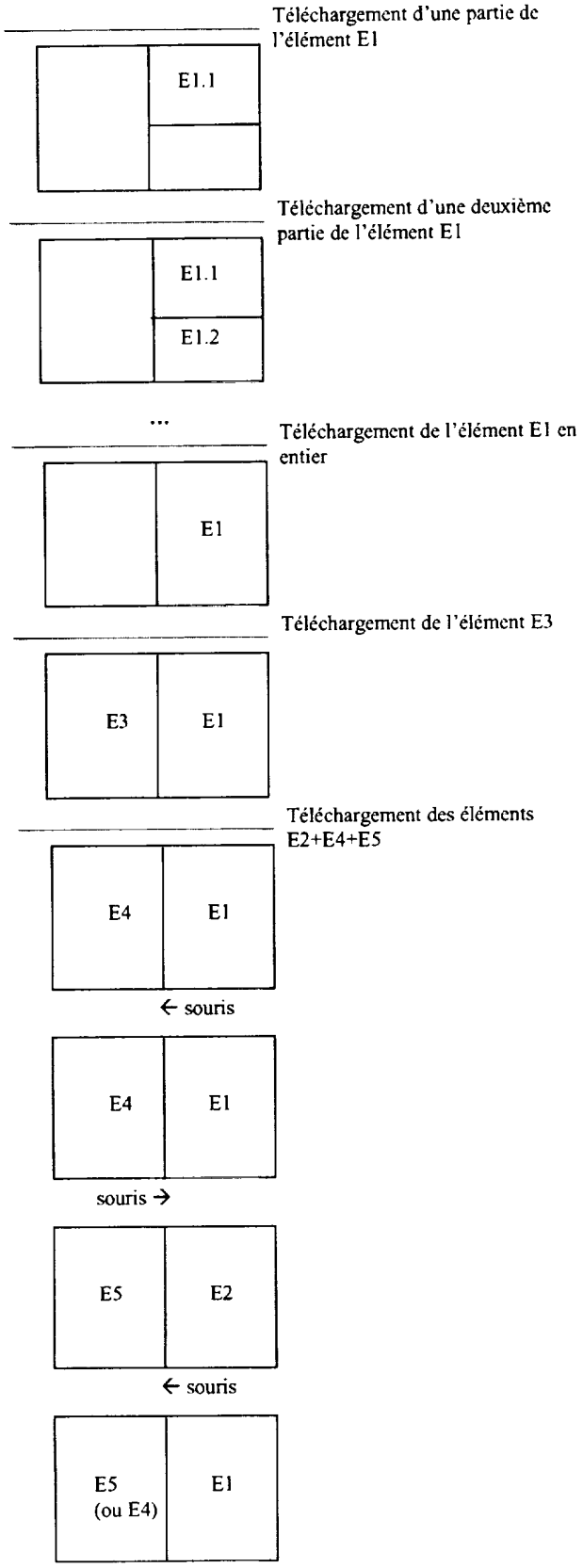

Téléchargement d'une partie de l'élément E1
E1.1
Téléchargement d'une deuxième partie de l'élément E1
E1.1
E1.2
...
Téléchargement de l'élément E1 en entier
E1
Téléchargement de l'élément E3
E3
E1
Téléchargement des éléments E2+E4+E5
E4
E1
← souris
E4
E1
souris →
E5
E2
← souris
E5
(ou E4)
E1

Fig. 13

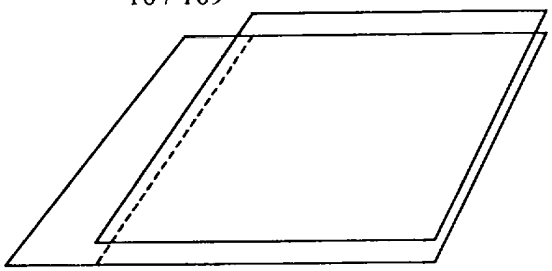


Fig.14

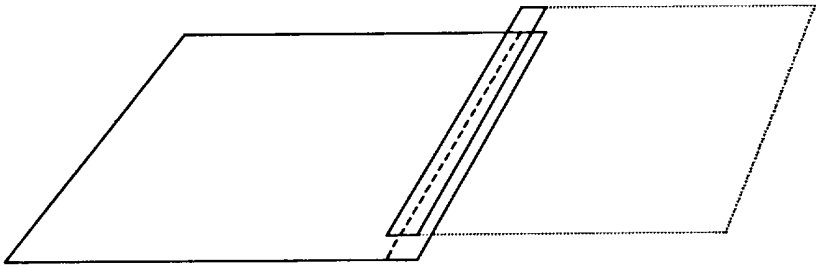


Fig.15

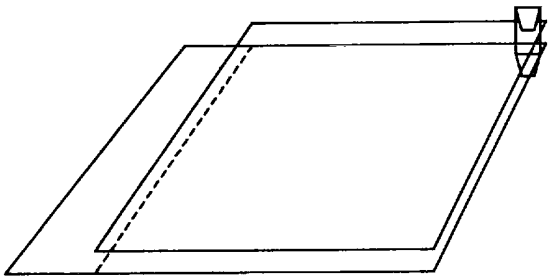


Fig.16

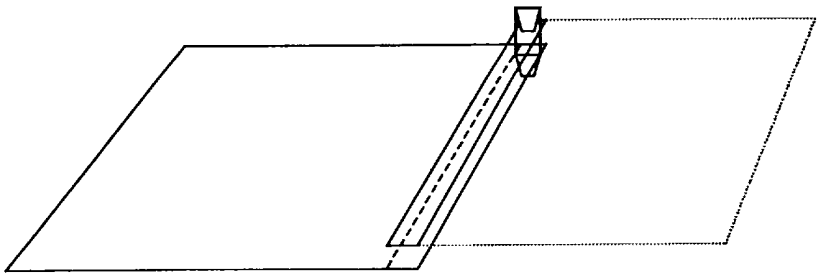


Fig.17

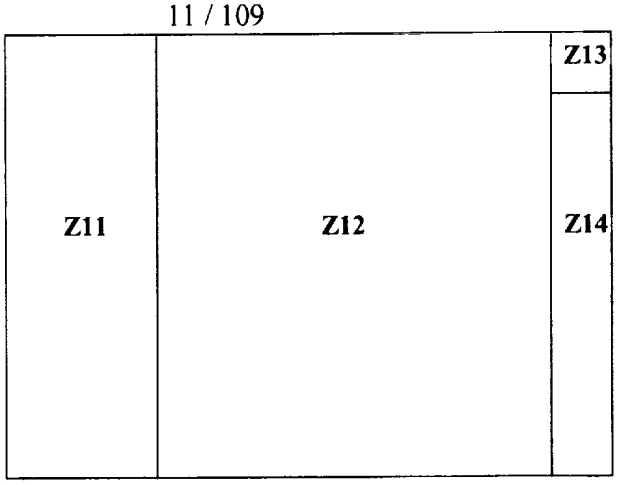


Fig.18

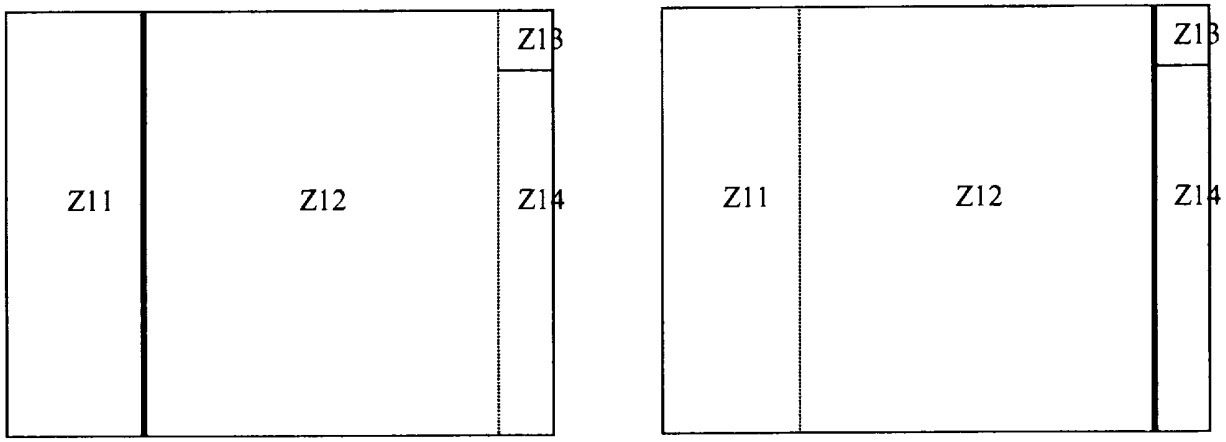


Fig.19

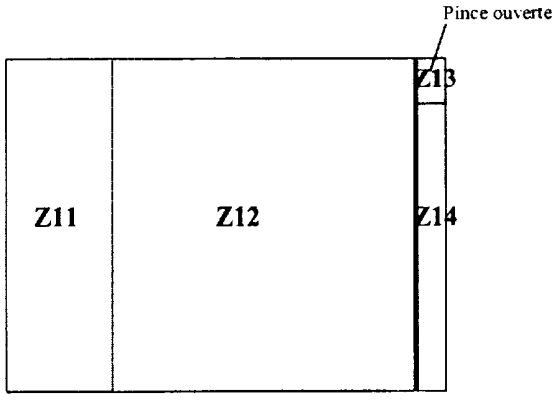


Fig.20

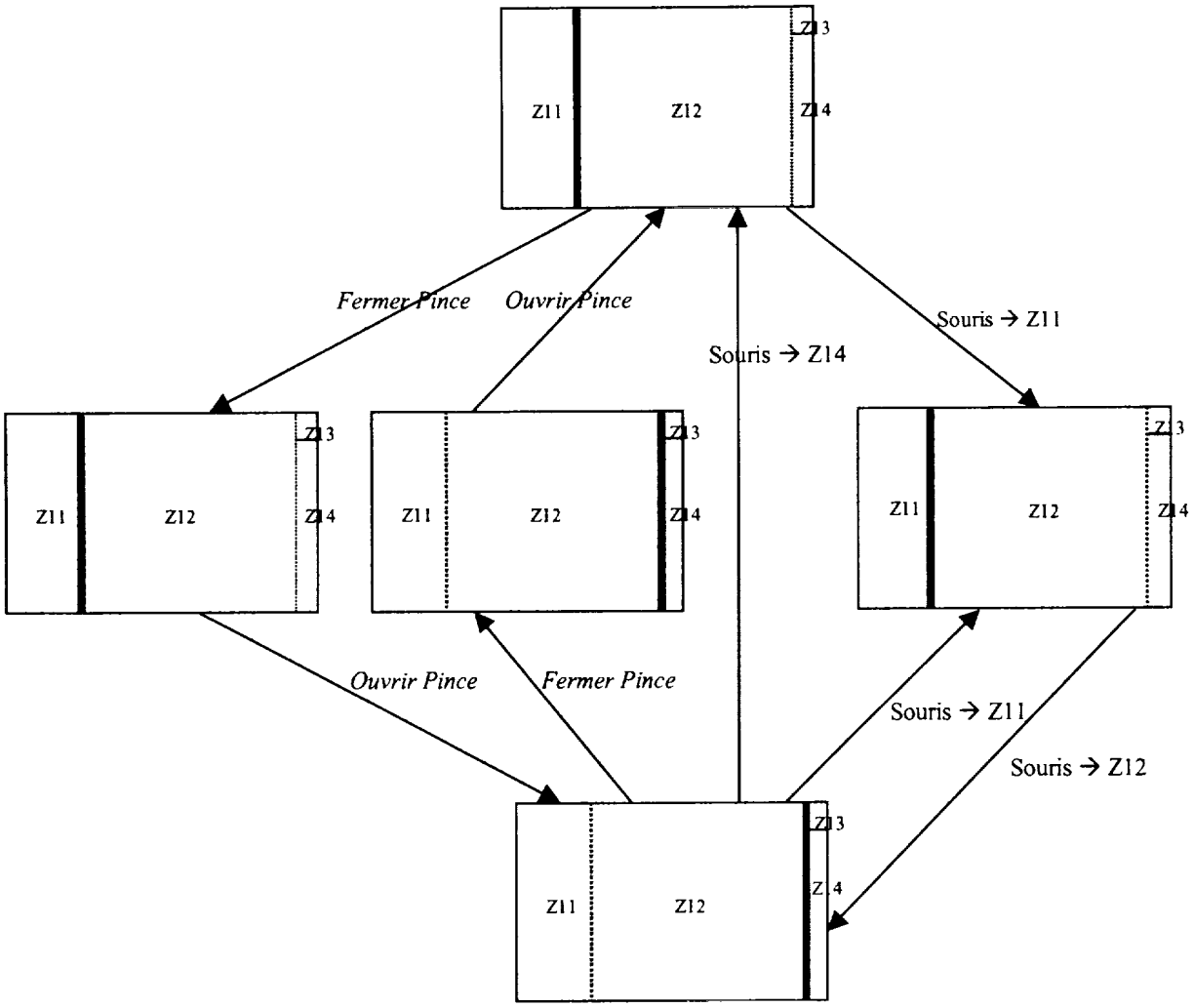
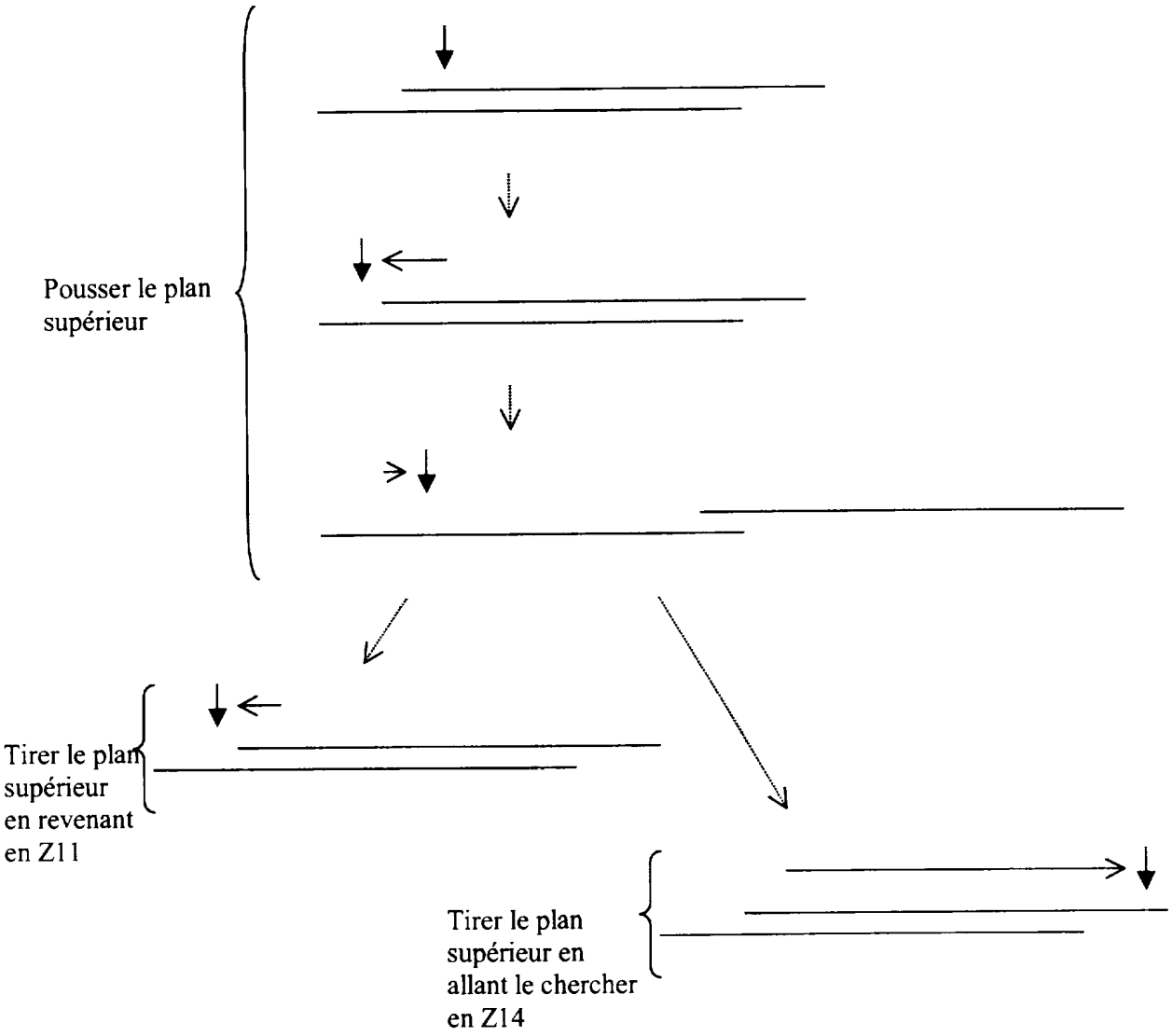

Z11
Z12
Z13
Z14
Fermer Pince
Ouvrir Pince
Souris → Z14
Souris → Z11
Ouvrir Pince
Fermer Pince
Souris → Z11
Souris → Z12

Fig.21



Légende

: Position de la souris

: Mouvement de la souris

: Plans vus en coupe

Fig.22

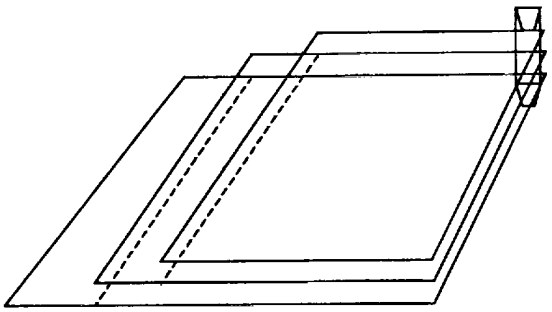


Fig.23

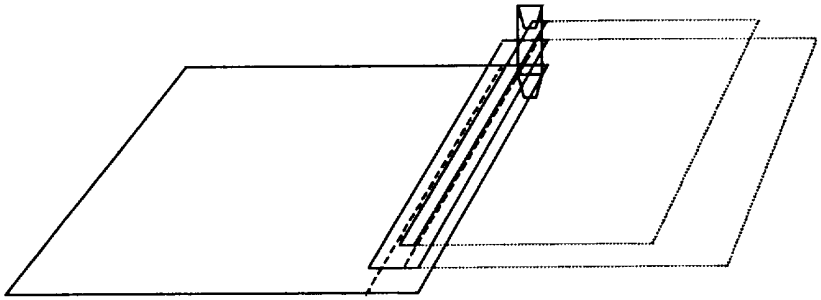


Fig.24

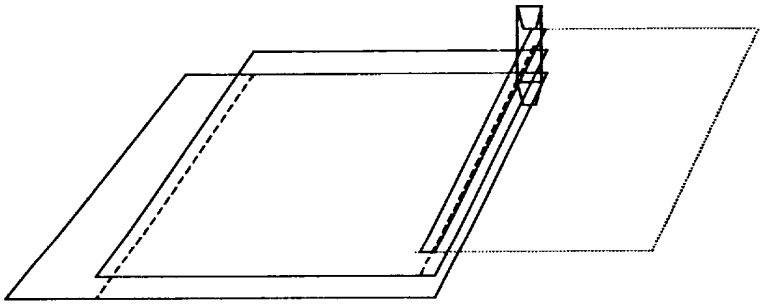
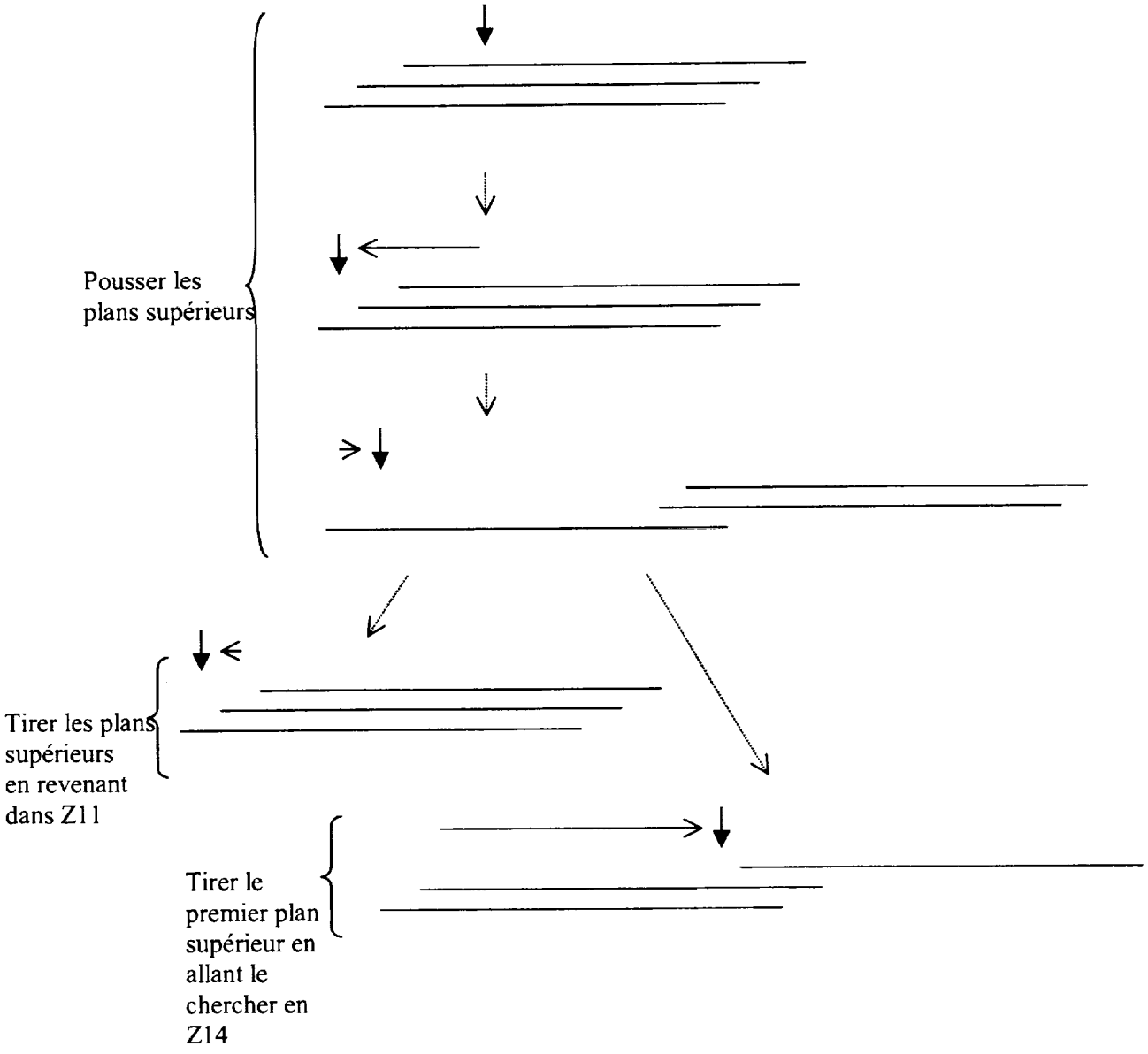


Fig.25



Légende

: Position de la souris

: Mouvement de la souris

: Plans vus en coupe

Fig.26

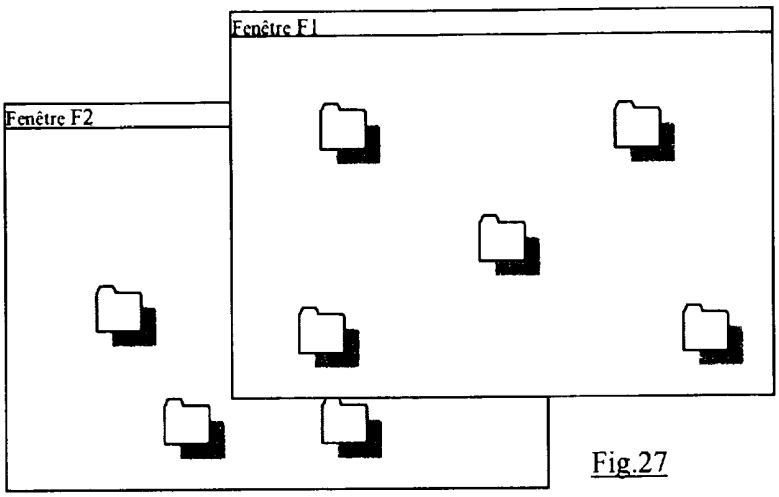


Fig.27

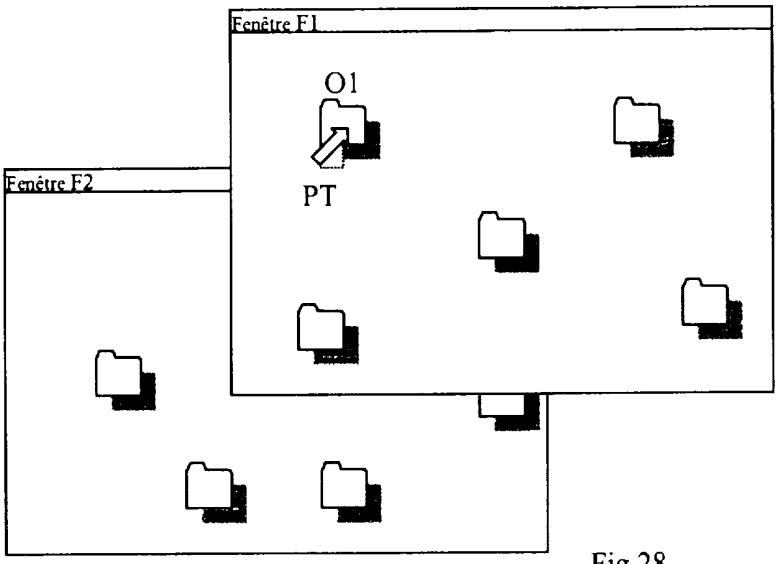


Fig.28

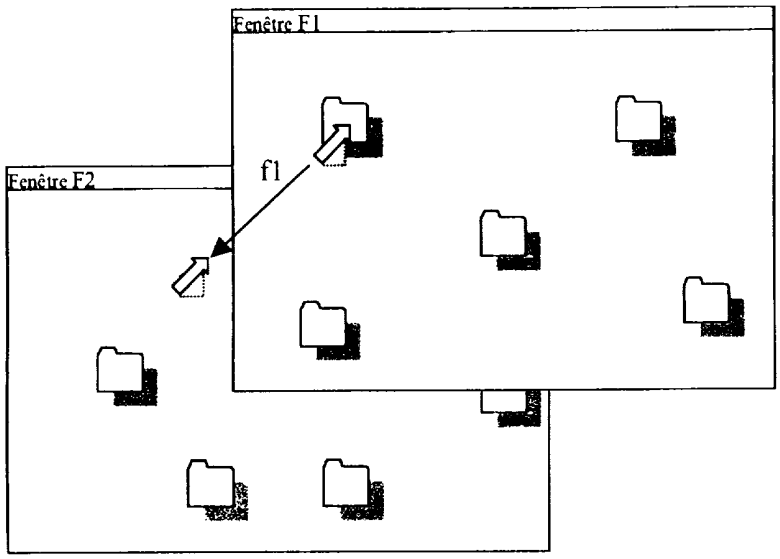


Fig.29

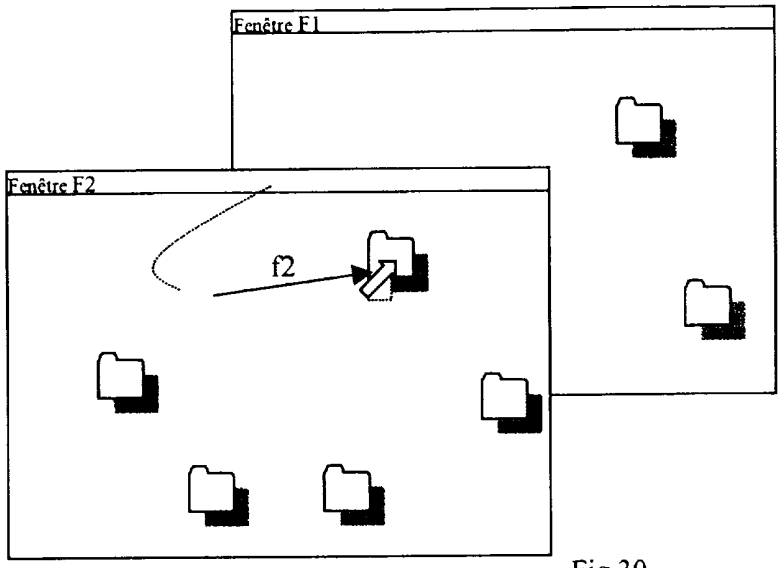


Fig.30

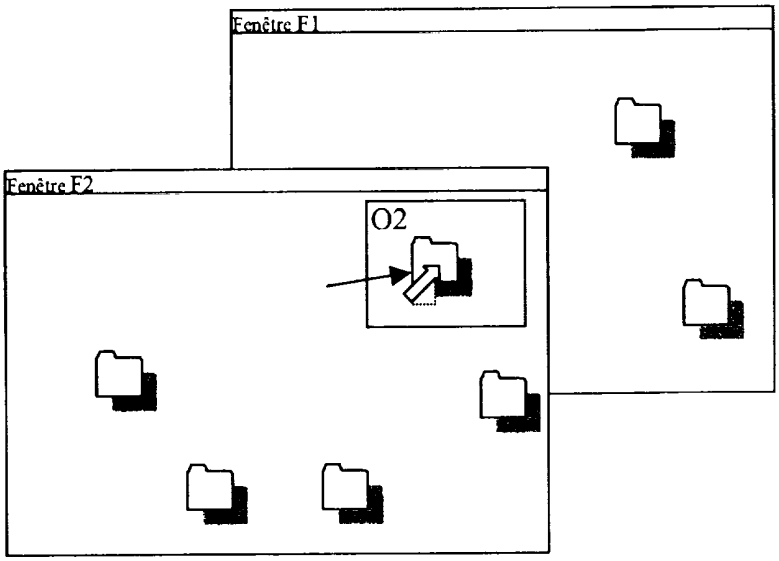
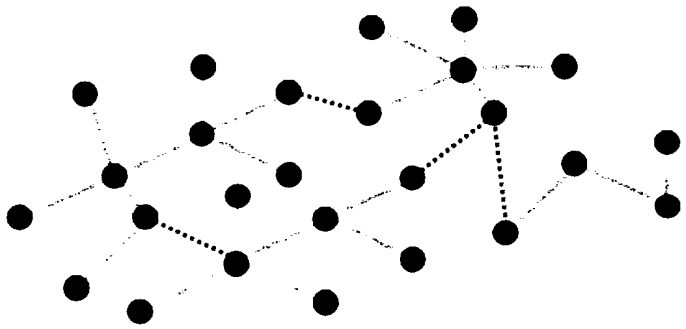


Fig.31



Web existant

Liens ajoutés par l'utilisateur

Fig.32

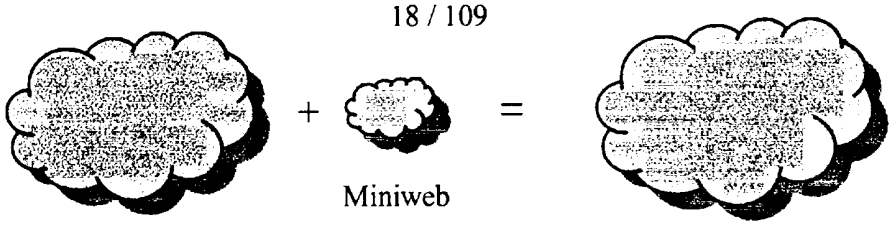


Fig.33

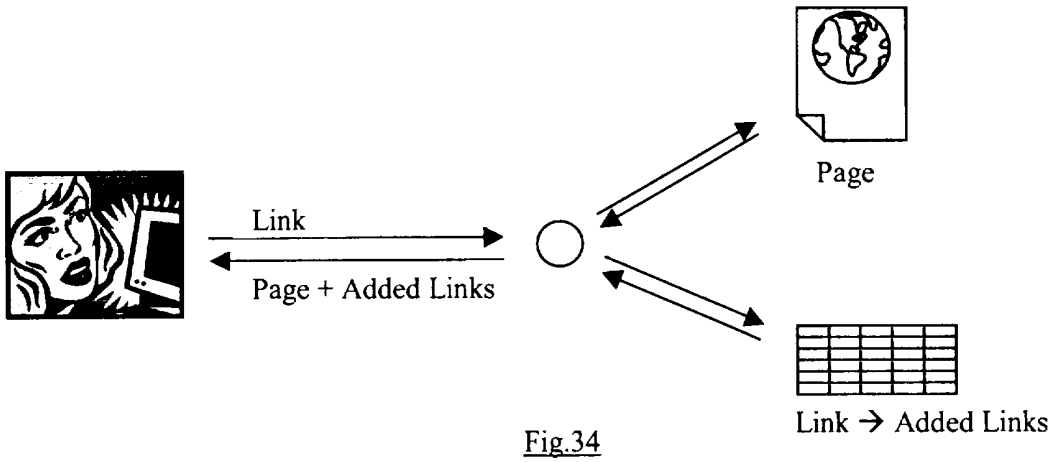
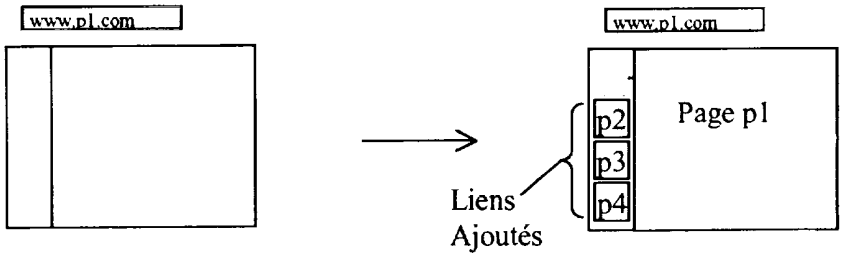


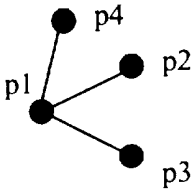
Fig.34



L'utilisateur accède à la page p1

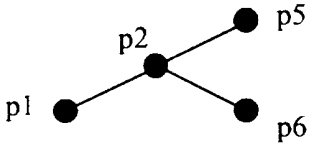
Les Liens Ajoutés à p1 apparaissent

Fig.35



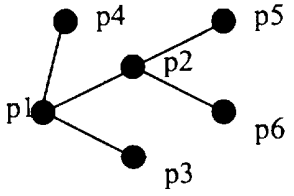
Pages pertinentes dans le contexte de la page p1.

Fig.36a



Pages pertinentes dans le contexte de la page p2.

Fig.36b



Pages pertinentes dans le contexte des pages p1 et p2 prises ensemble.

Fig.36c

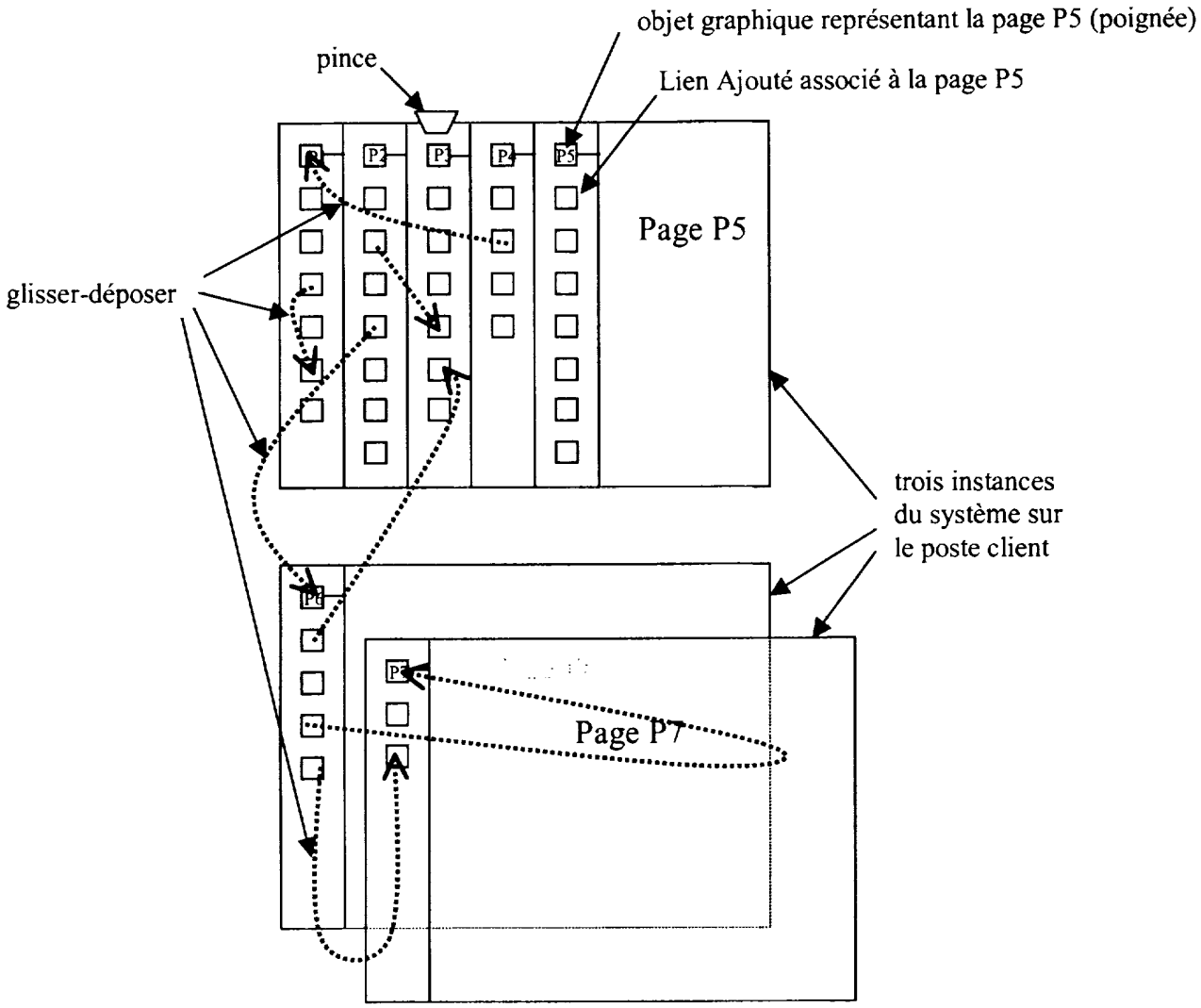

objet graphique représentant la page P5 (poignée)
pince
Lien Ajouté associé à la page P5
P1
P2
P3
P4
P5
Page P5
glisser-déposer
trois instances
du système sur
le poste client
P6
P7
Page P7

Fig.37

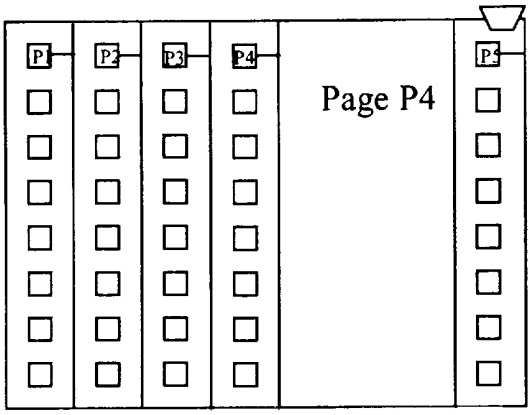


Fig.38

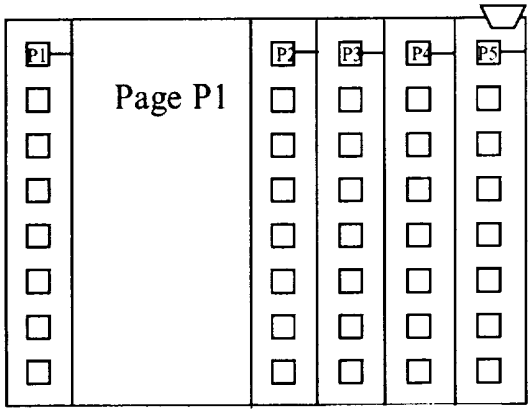


Fig.39

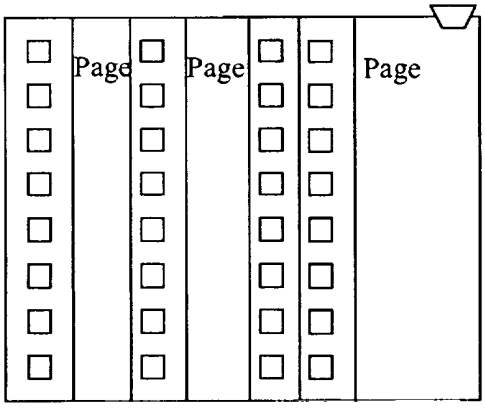


Fig.40

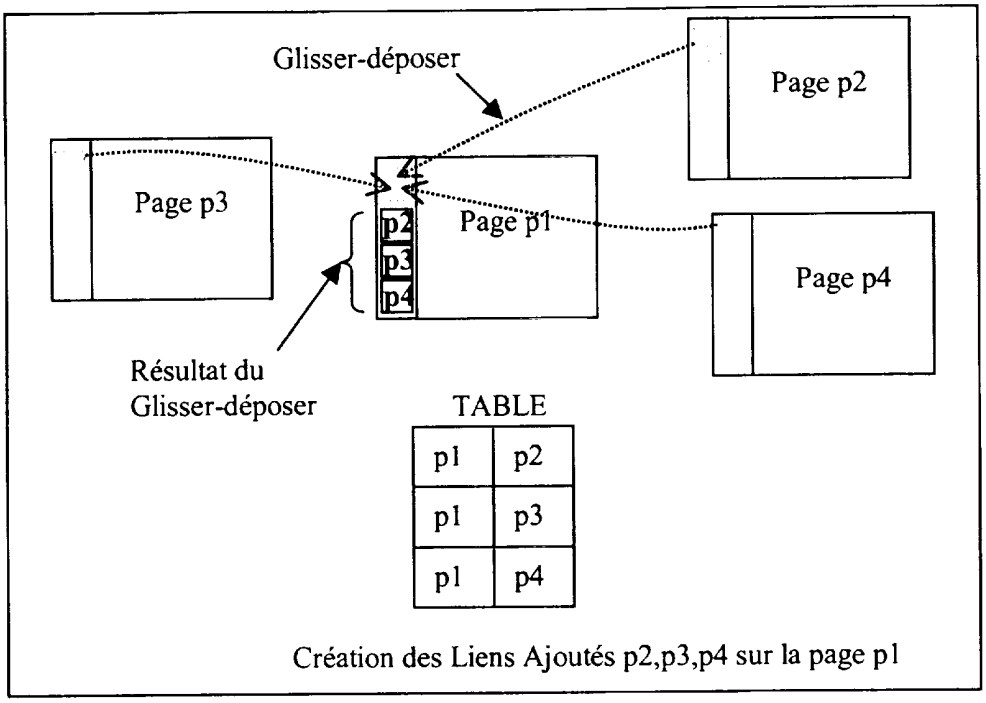

Glisser-déposer
Page p2
Page p3
p2
p3
p4
Page p1
Page p4
Résultat du
Glisser-déposer
TABLE
p1 p2
p1 p3
p1 p4
Création des Liens Ajoutés p2,p3,p4 sur la page p1

Fig.41

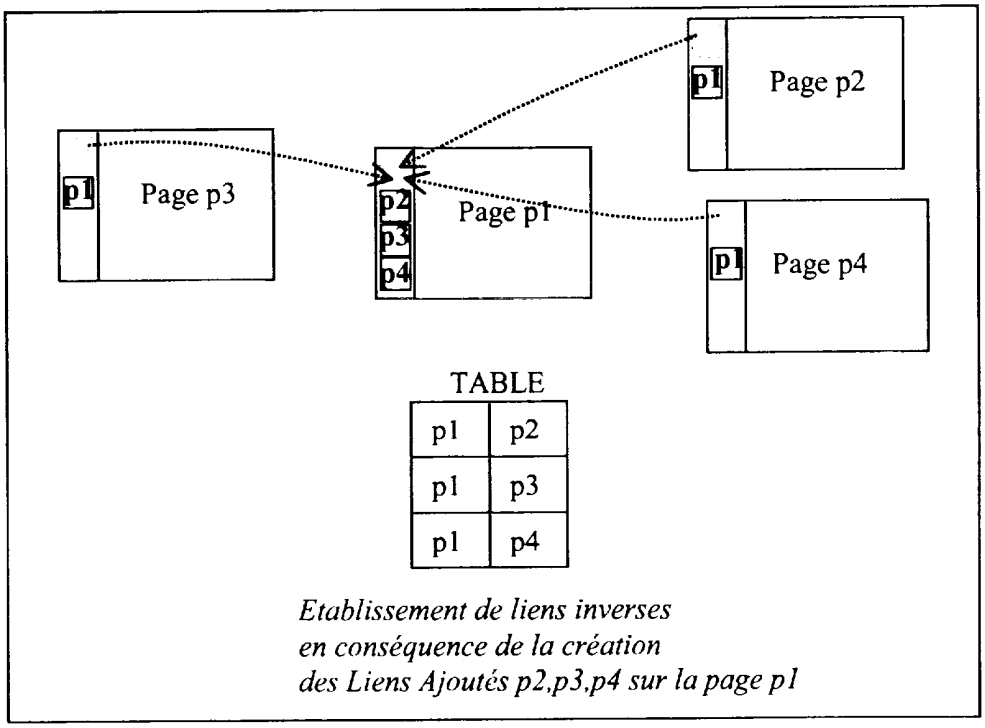

p1
Page p2
p1
Page p3
p2
p3
p4
Page p1
p1
Page p4
TABLE
p1 p2
p1 p3
p1 p4
Etablissement de liens inverses
en conséquence de la création
des Liens Ajoutés p2,p3,p4 sur la page p1

Fig.42

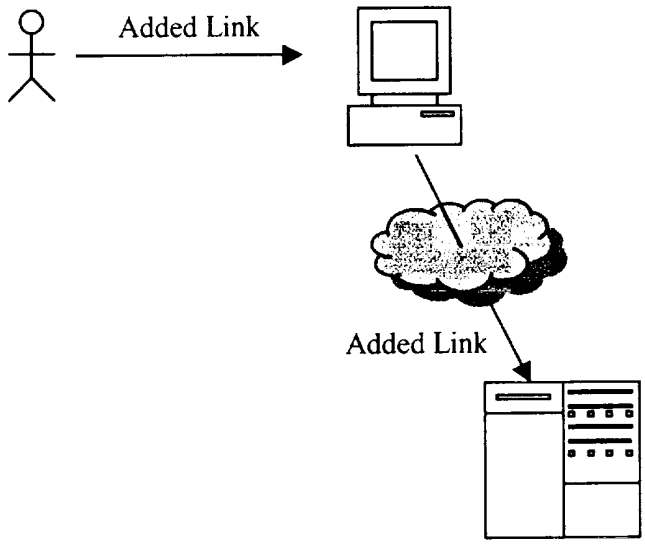

Added Link
Added Link

Fig.43

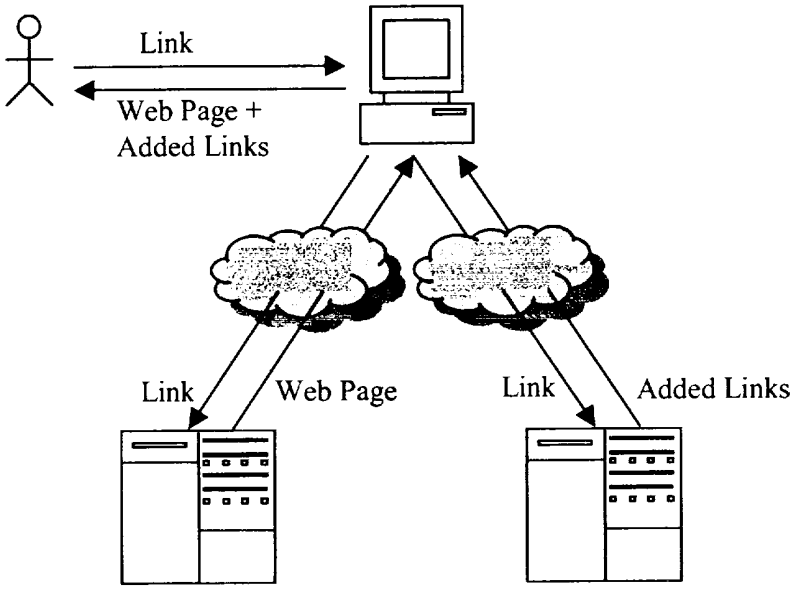

Link
Web Page +
Added Links
Link
Web Page
Link
Added Links

Fig.44

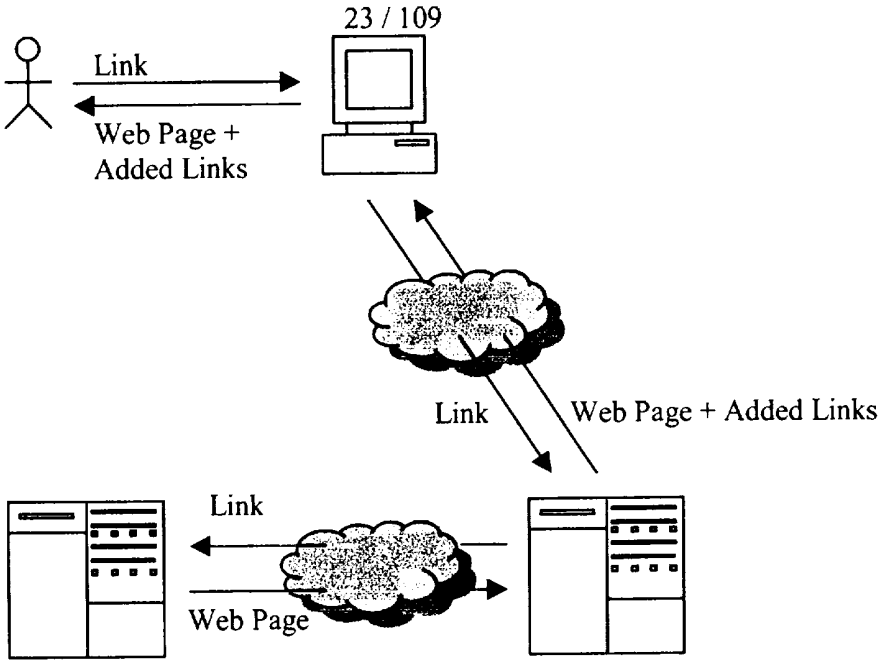
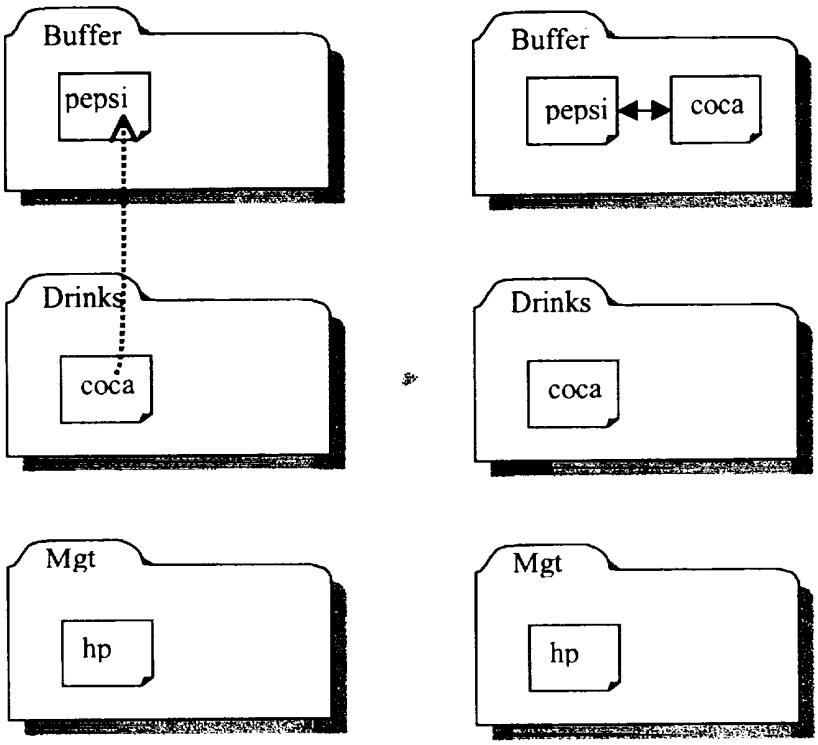


Fig.45



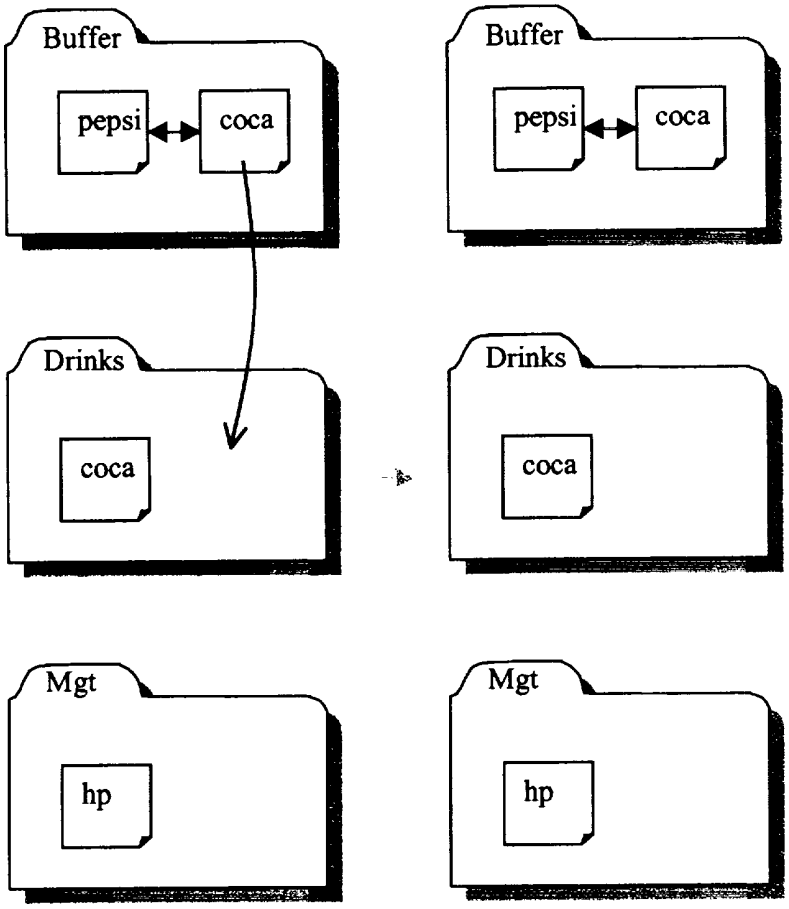
: Glisser-déposer une page pour tirer un Lien Ajouté

coca : Page (en fait il s'agit d'un lien sur une page)

: Lien Ajouté (bidirectionnel) entre deux pages d'un répertoire

« coca », « pepsi » et « hp » sont des marques déposées

Fig.46

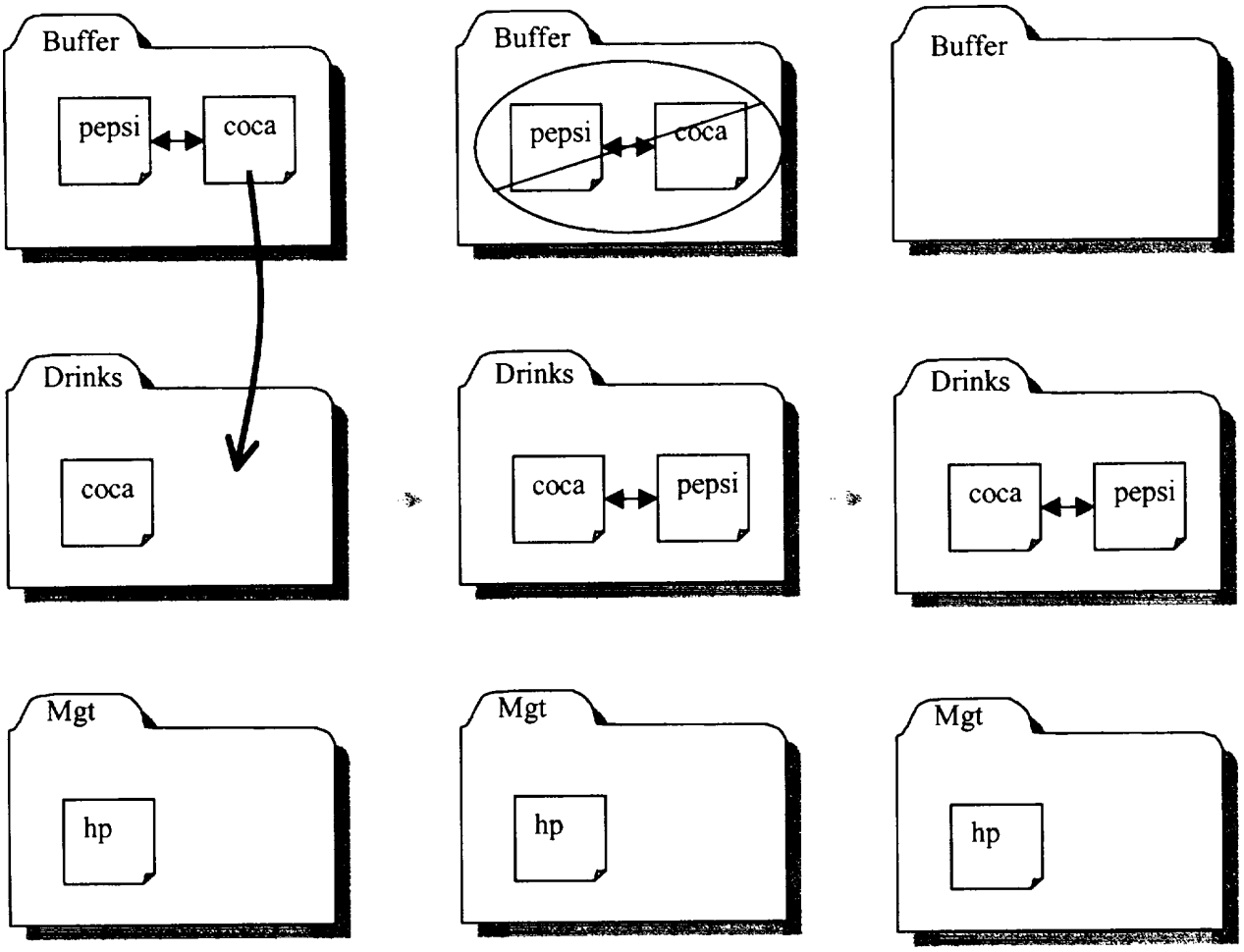


: Déplacement de page (en fait il s'agit plutôt de déplacer un lien vers une page)

: Copie de page (en fait il s'agit plutôt de copier un lien vers une page)

« coca », « pepsi » et « hp » sont des marques déposées

Fig.47



: Déplacement de page (en fait il s'agit plutôt de déplacer un lien vers une page) avec tous ses liens

: Copie de page (en fait il s'agit plutôt de copier un lien vers une page) avec tous ses liens

: Suppression d'un ensemble de (liens vers des) pages

« coca », « pepsi » et « hp » sont des marques déposées

Fig.48

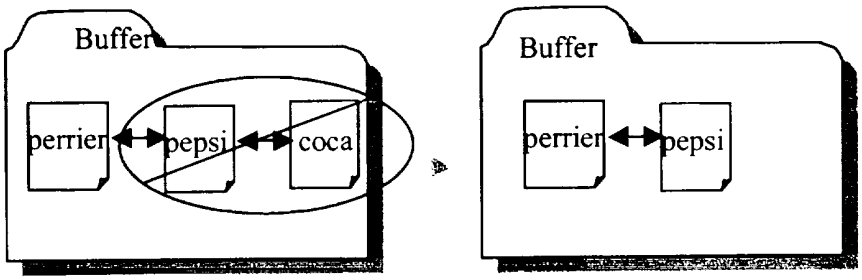
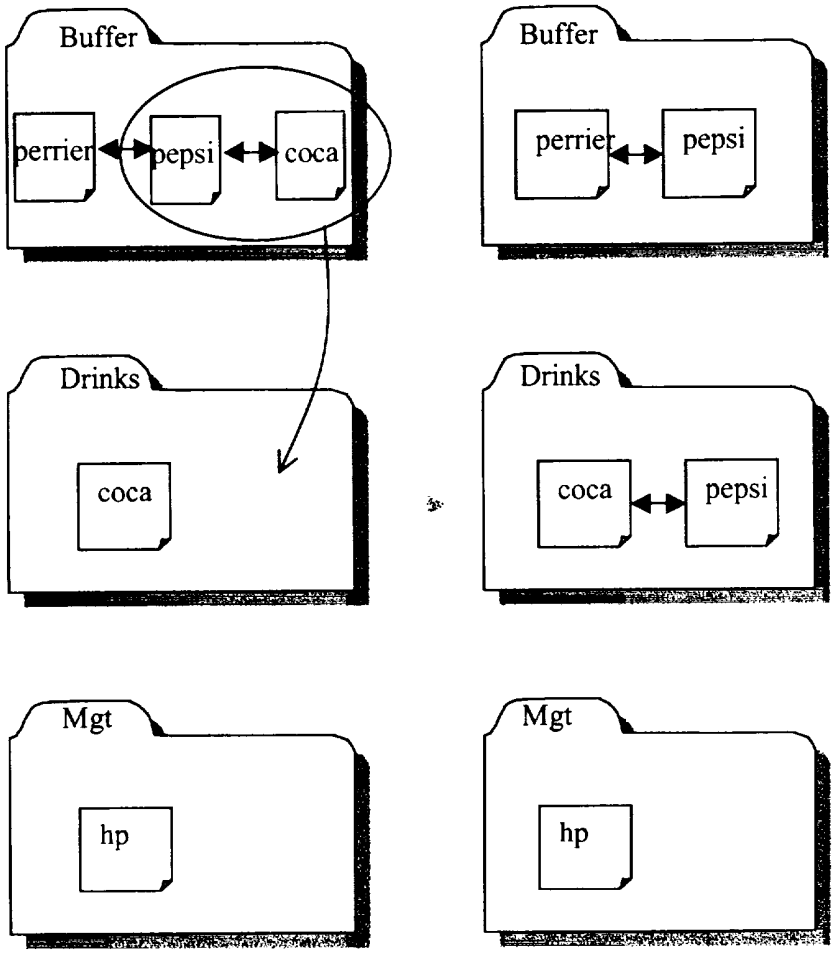


Fig.49

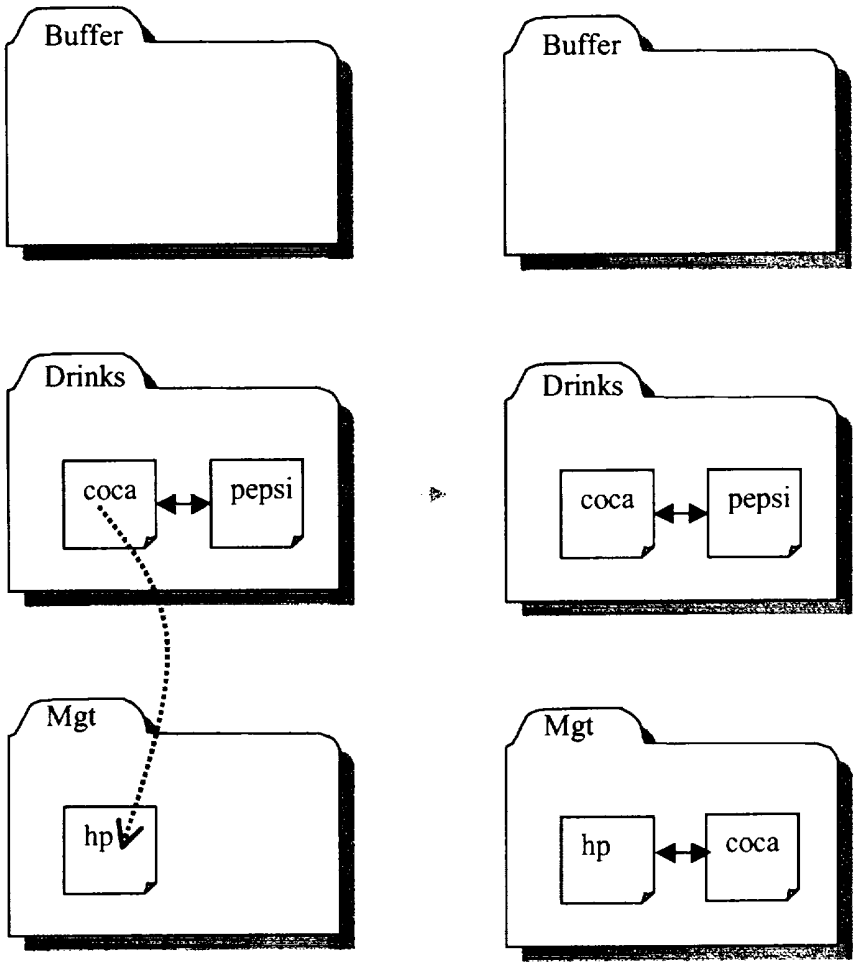


: Déplacement d'un ensemble de pages

: Copie d'un ensemble de pages

« perrier », « coca », « pepsi » et « hp » sont des marques déposées

Fig.50



: Glisser-déposer une page pour tirer un Lien Ajouté

« coca », « pepsi » et « hp » sont des marques déposées

Fig.51

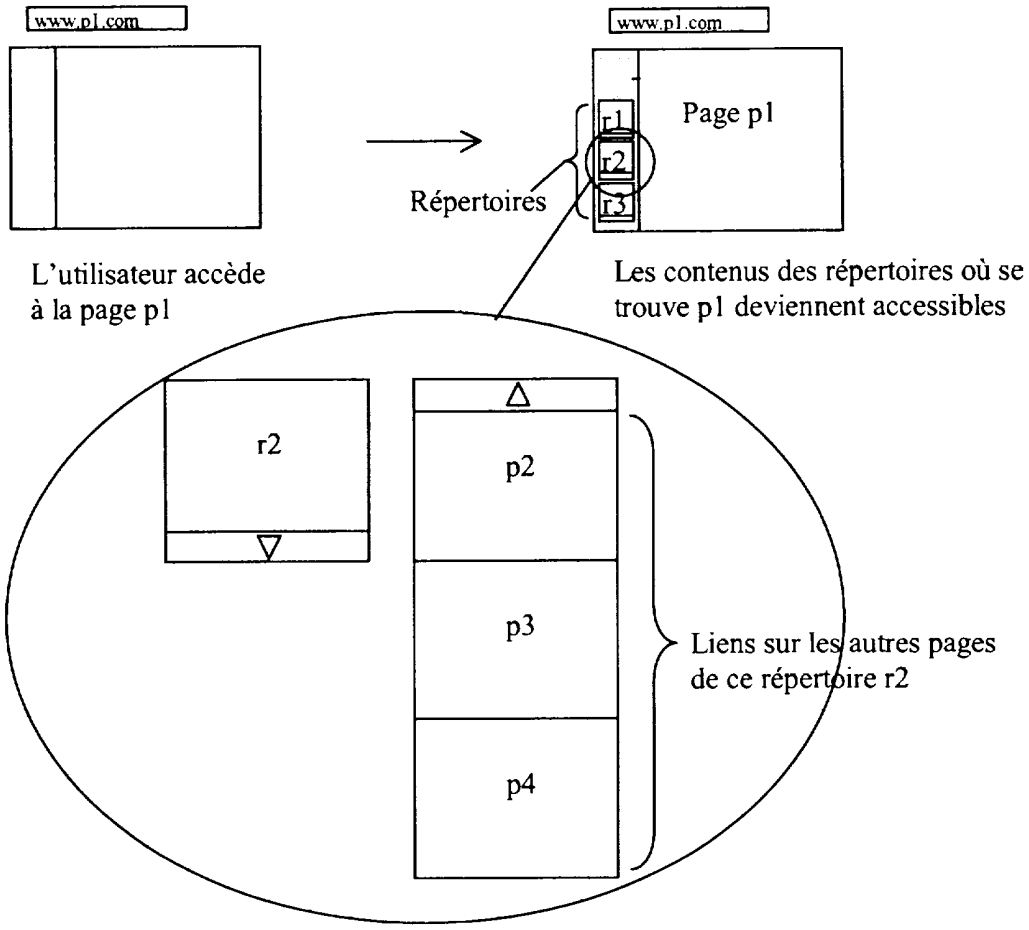

www.p1.com
www.p1.com
Page p1
r1
r2
r3
Répertoires
L'utilisateur accède à la page p1
Les contenus des répertoires où se trouve p1 deviennent accessibles
r2
p2
p3
p4
Liens sur les autres pages de ce répertoire r2

Fig.52

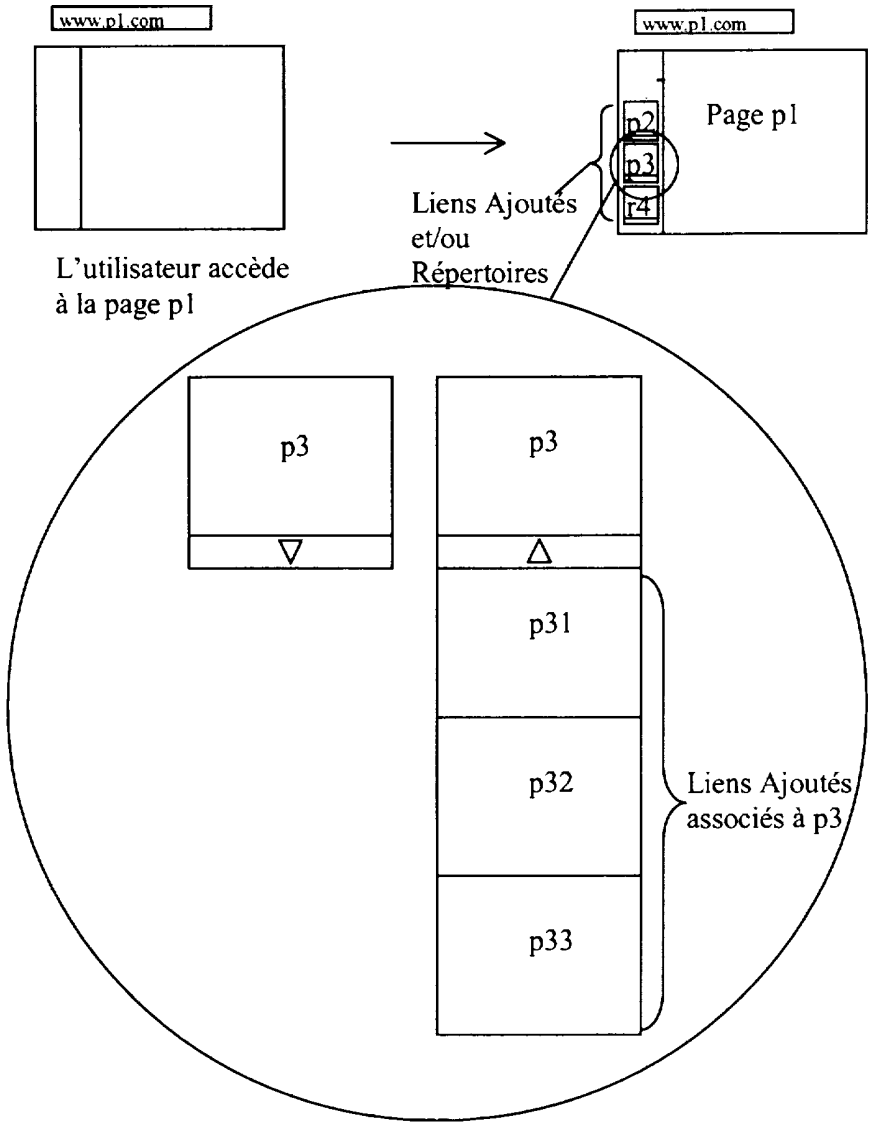
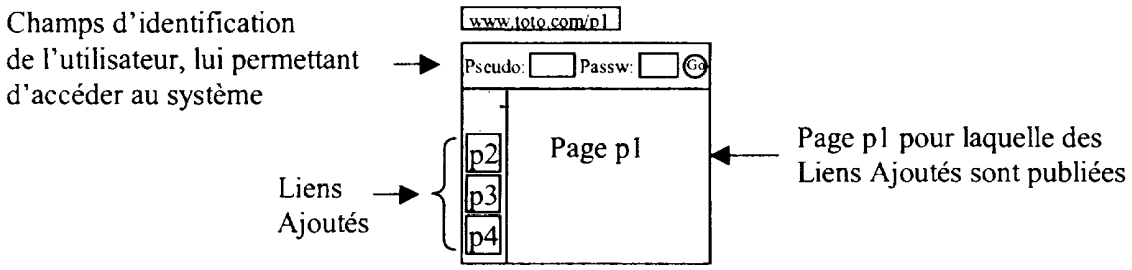


Fig.53



Liens Ajoutés communiqués ou publiés de manière « classique »
et invitant à utiliser le système

Fig.54

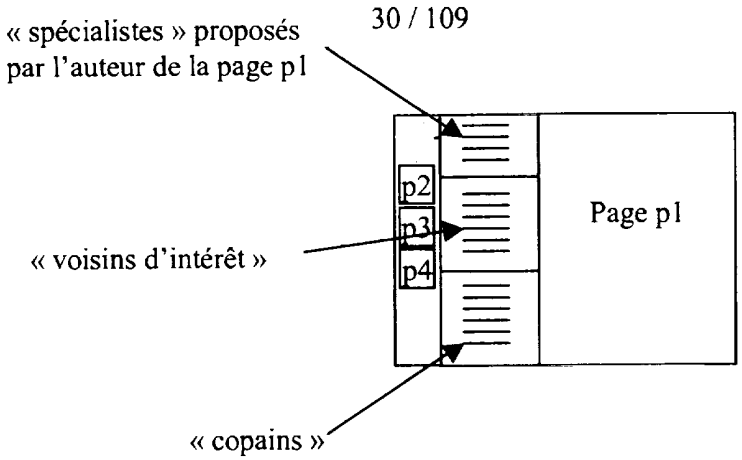


Fig.55a

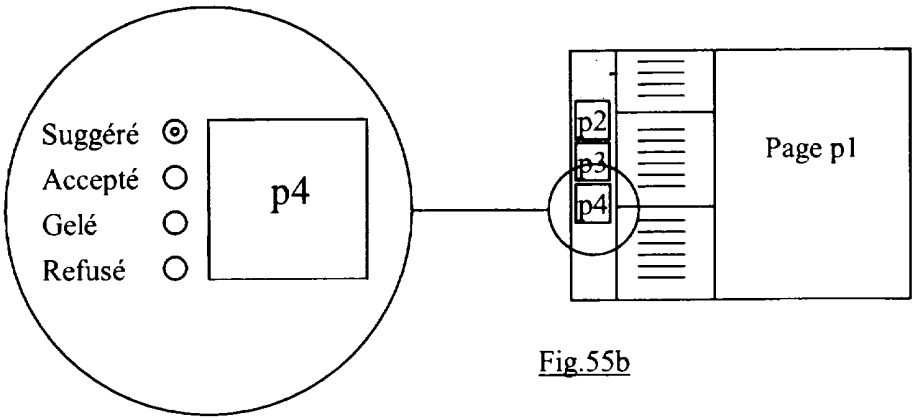


Fig.55b

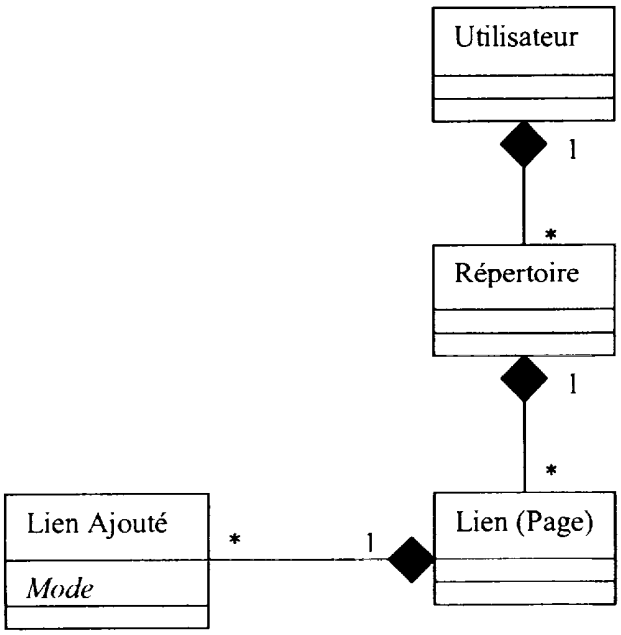
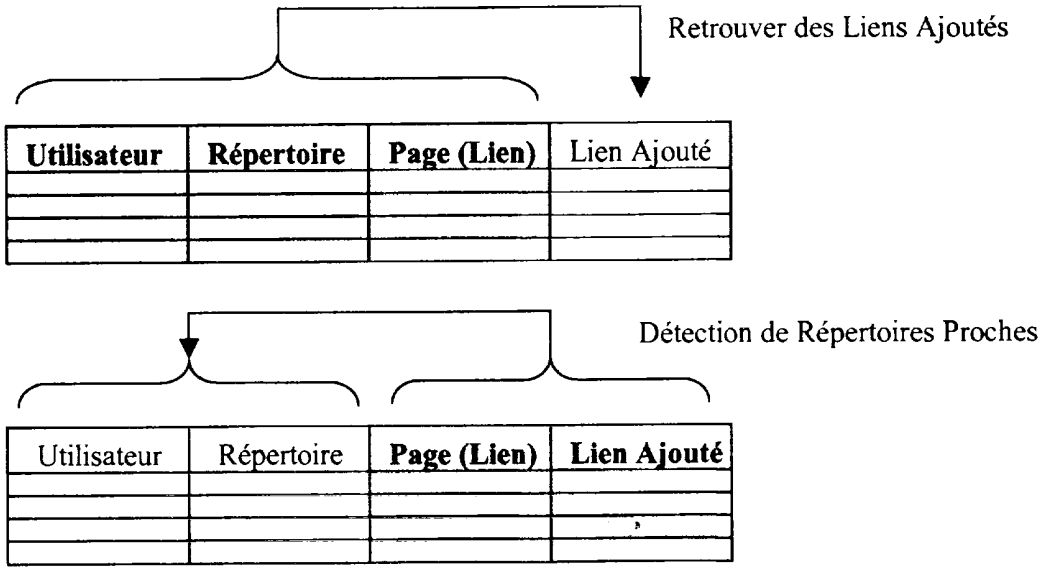
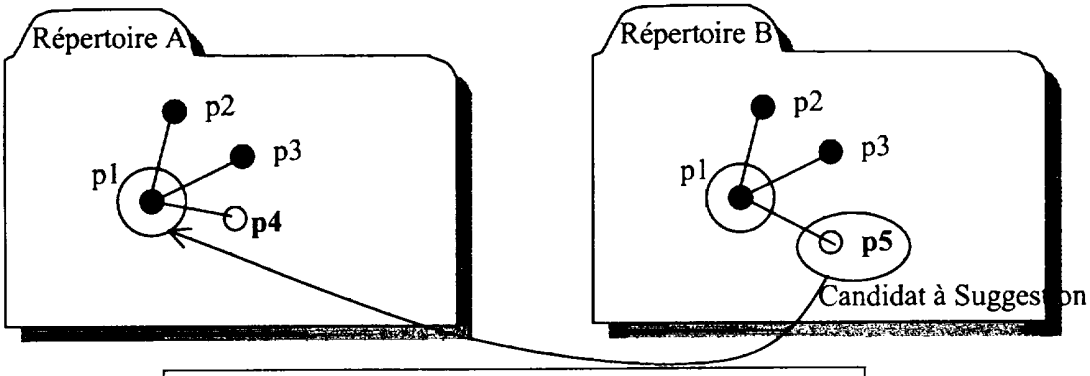


Fig.56



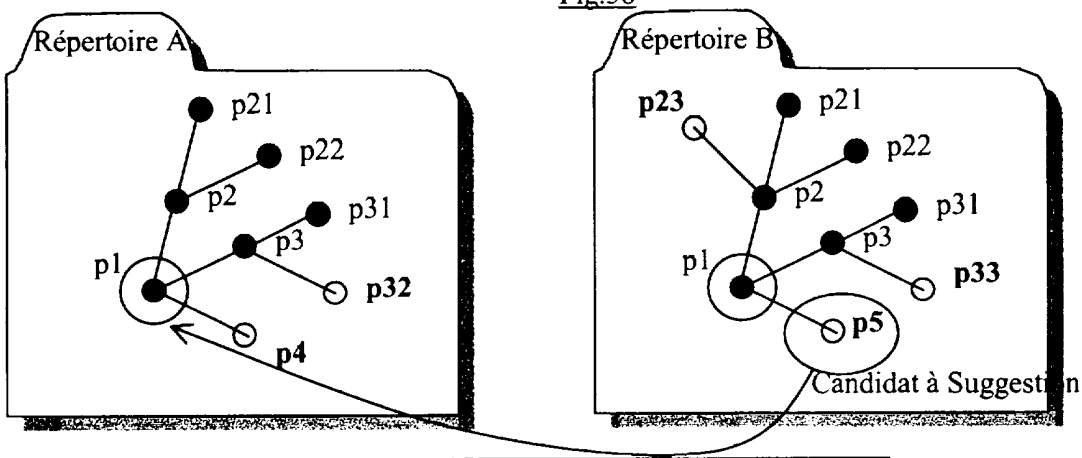
: Colonnes indéxées à partir desquelles s'effectue l'accès direct

Fig.57



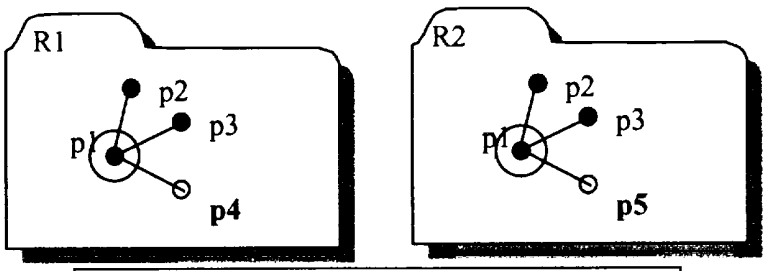
Proximité A-B au Niveau 1: 2 Liens Ajoutés sur 3

Fig.58

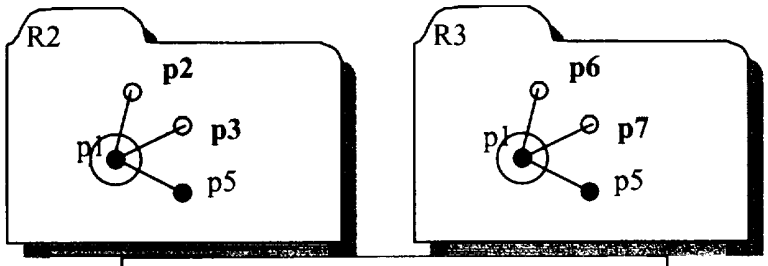


Proximité A-B au Niveau 2: 5 Liens Ajoutés sur 7

Fig.59



Proximité R1-R2: 2 Liens Ajoutés sur 3

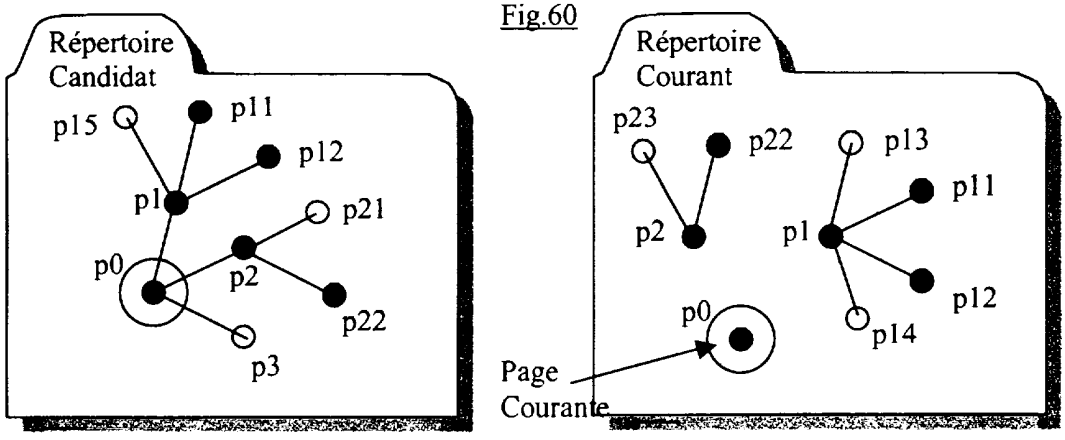


Proximité R2-R3: 1 Lien Ajouté sur 3

Proximité R1-R3: 0 Lien Ajouté sur 3

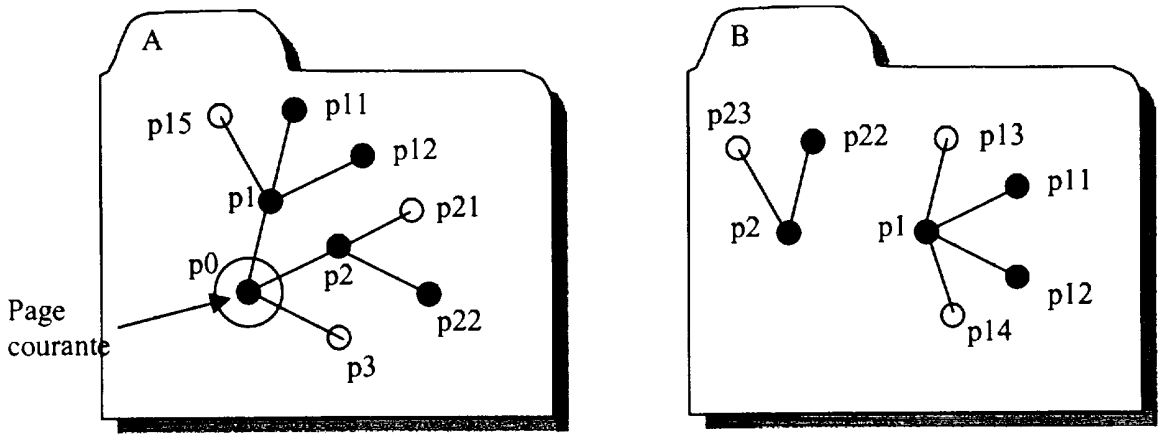
Proximité Transitive R1-R3: 1 Lien Ajouté sur 3

Fig.60



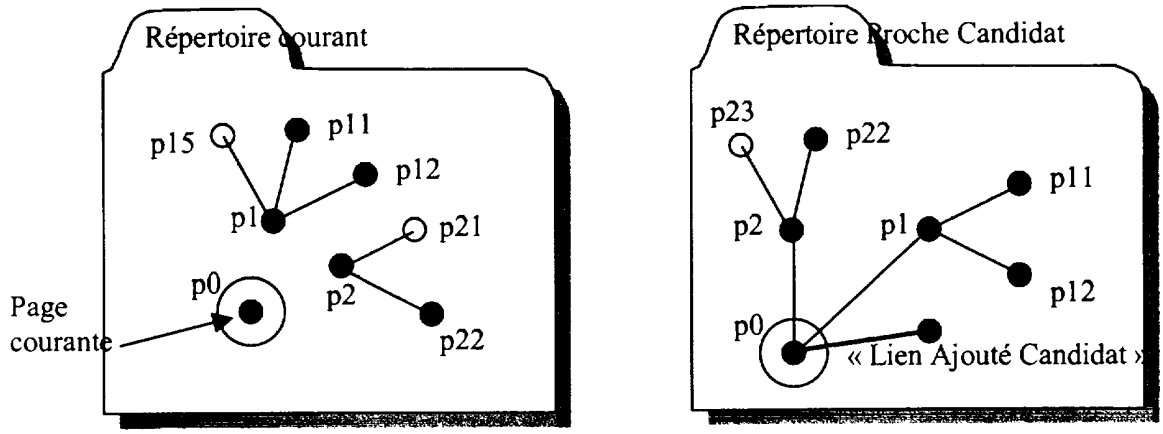
Proximité Décalée (éloignement 1):
1 fois 2 Liens Ajoutés sur 3
1 fois 1 Lien Ajouté sur 2

Fig.61



Proximité Décalée (éloignement 1):
1 fois 2 Liens Ajoutés sur 3
1 fois 1 Lien Ajouté sur 2

Fig.62



Proximité Décalée (éloignement 1):
1 fois 2 Liens Ajoutés sur 3
1 fois 1 Lien Ajouté sur 2

Fig.63

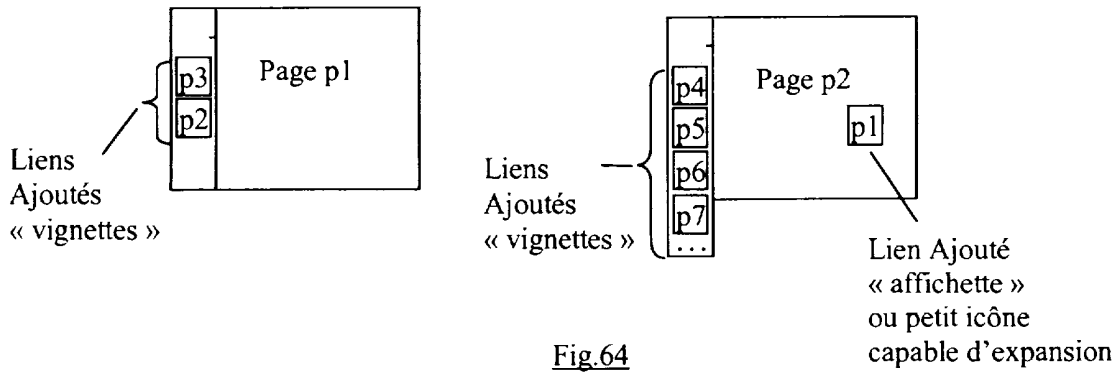
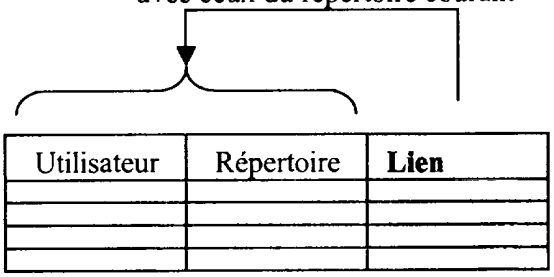


Fig.64

Sélection de Répertoires
dont une grande partie des Liens
sont en commun
avec ceux du répertoire courant



: Colonne indéxée à partir desquelles s'effectue l'accès direct

Fig.65

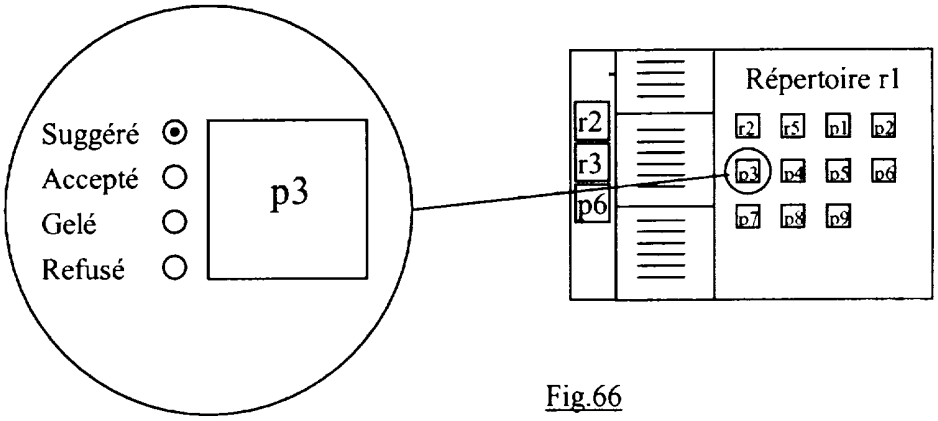


Fig.66

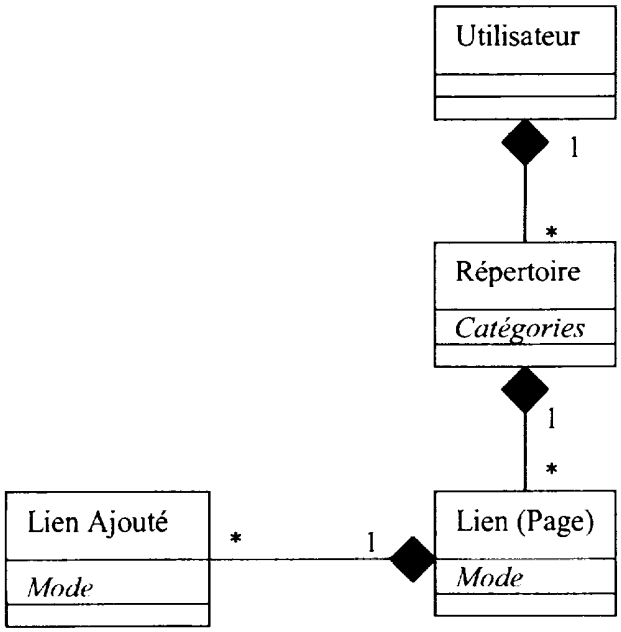


Fig.67

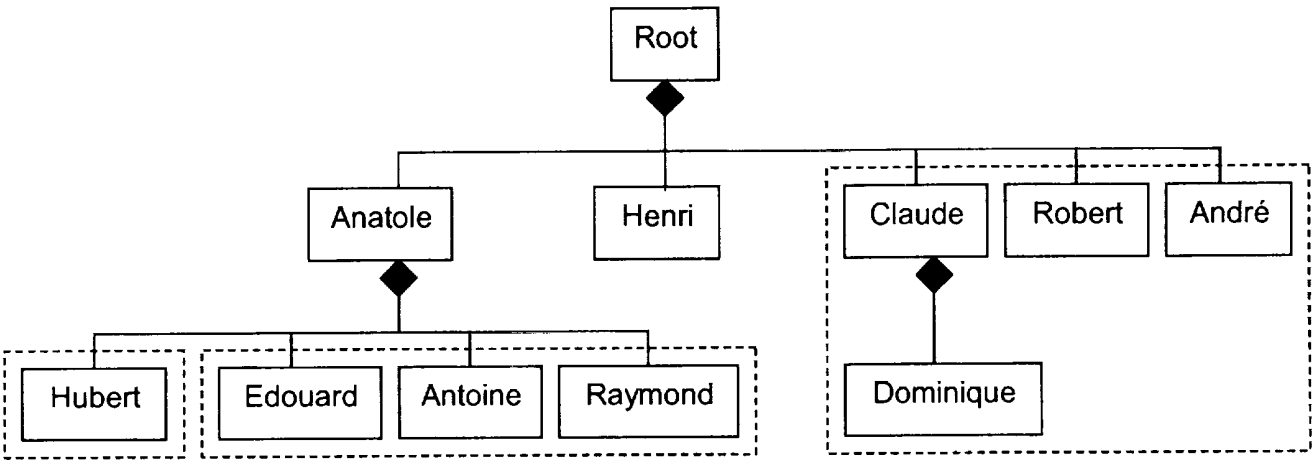


Fig.68

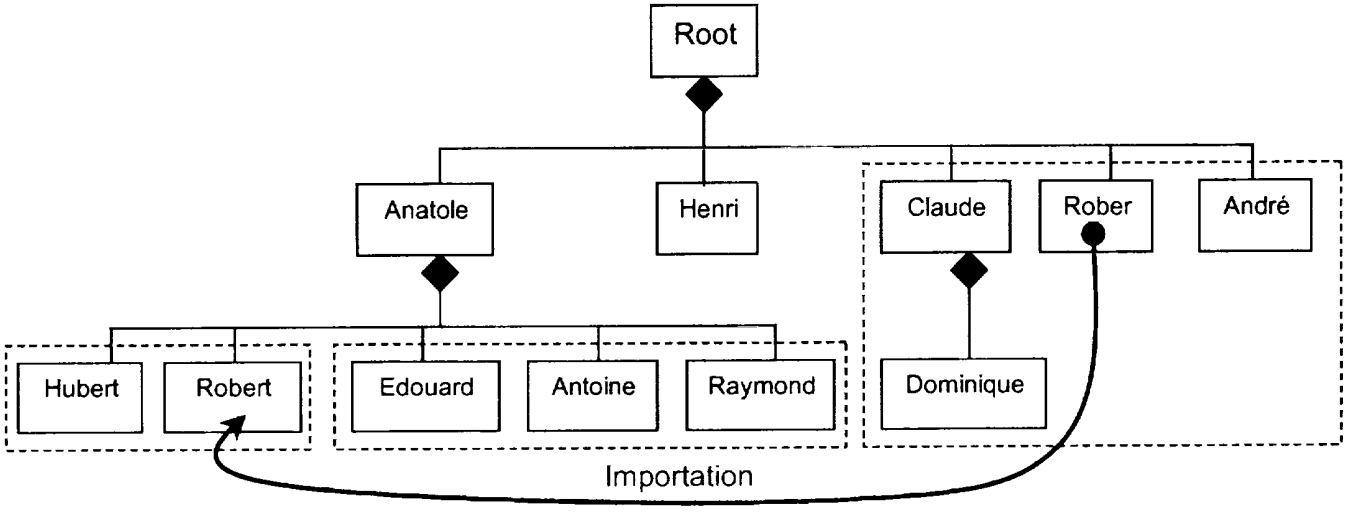


Fig.69

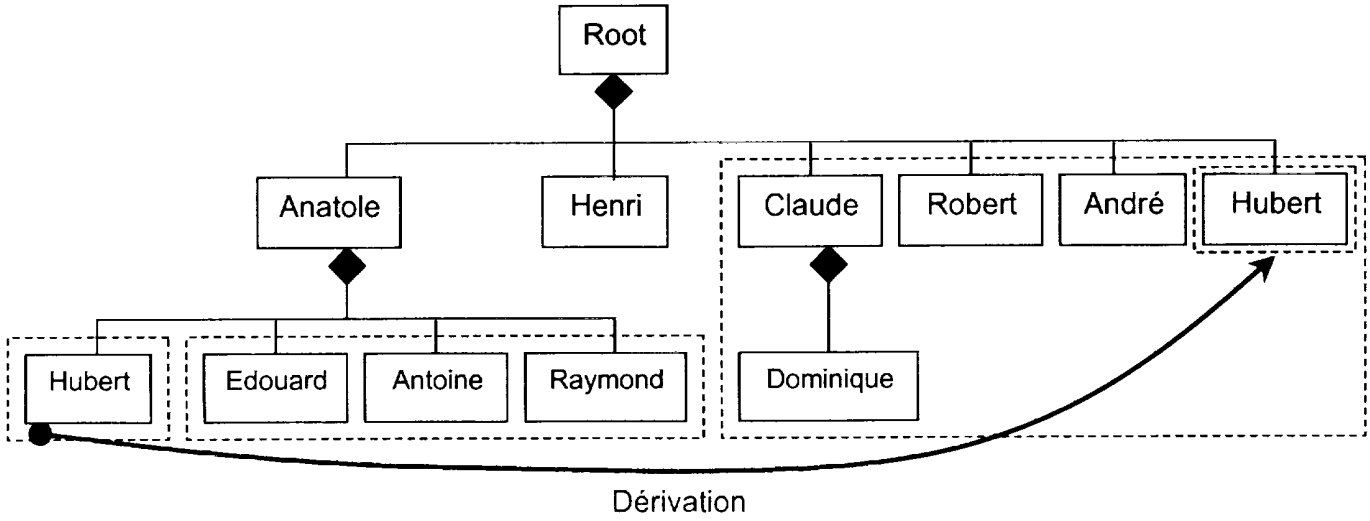


Fig.70

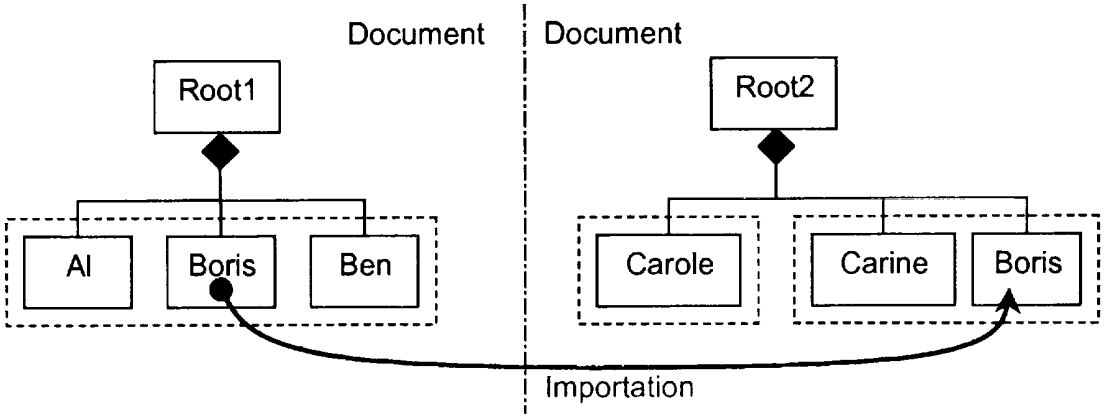


Fig.71

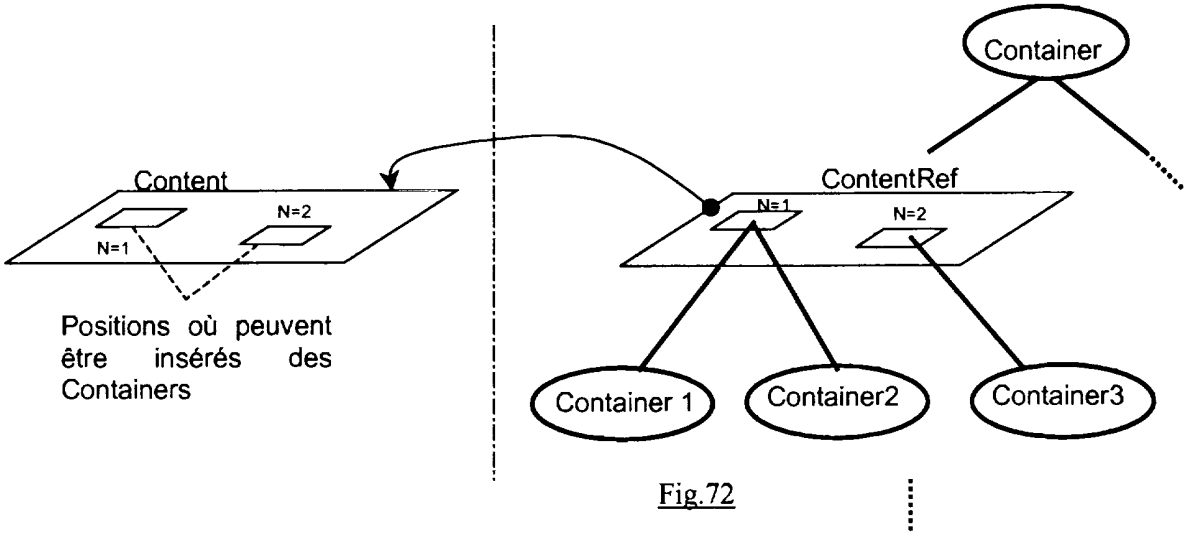


Fig.72

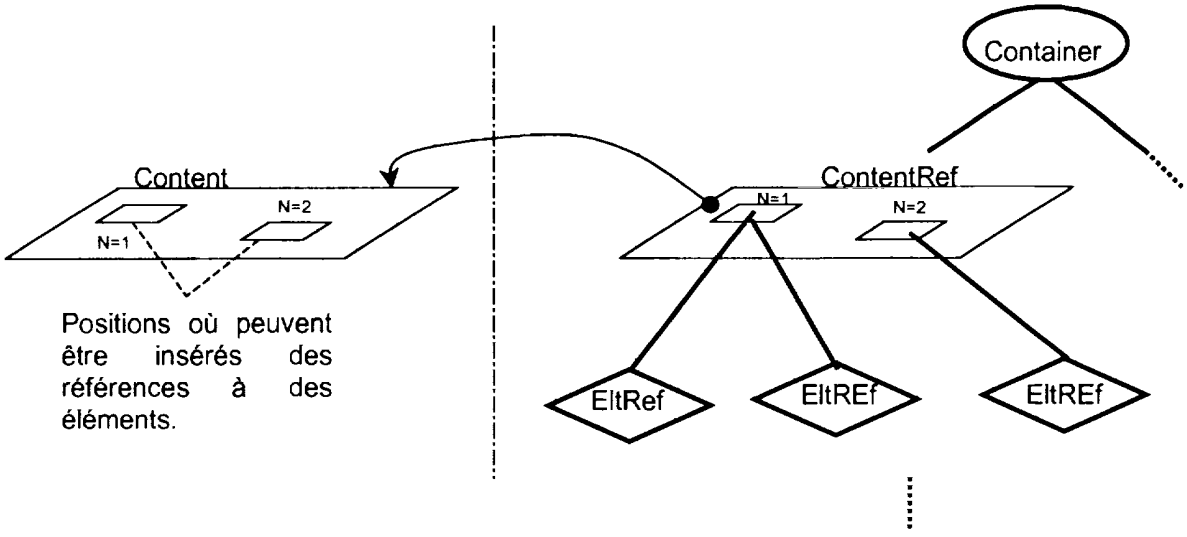


Fig.73

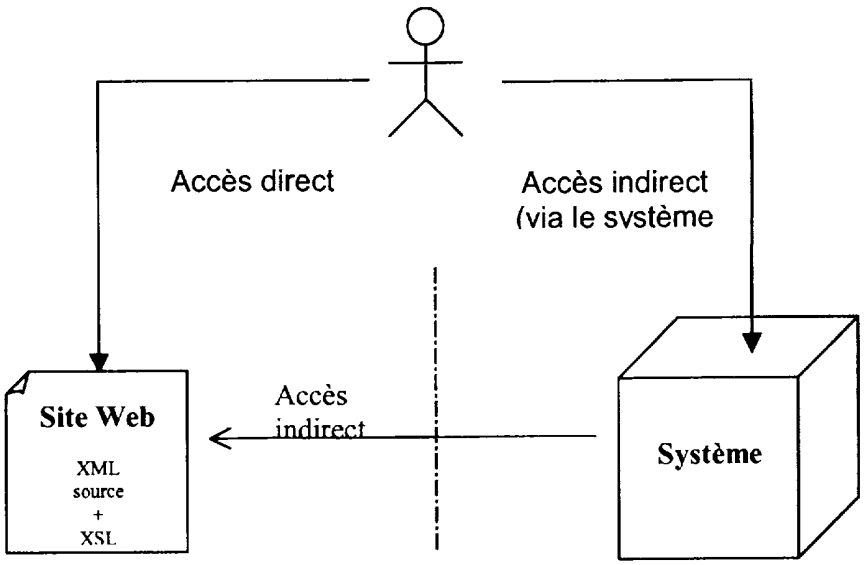

Accès direct
Accès indirect
(via le système
Site Web
XML
source
+
XSL
Accès
indirect
Système

Fig.74

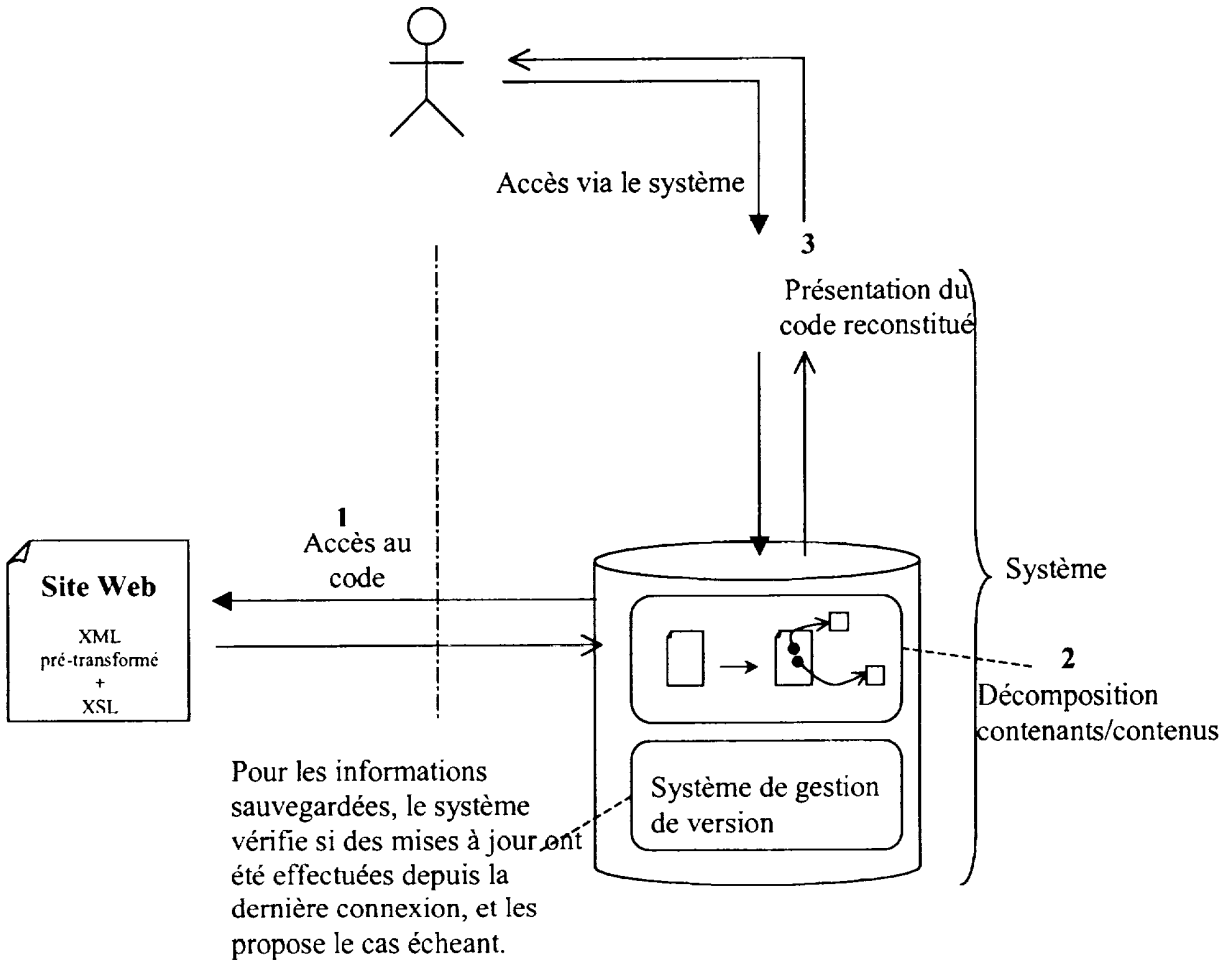

Accès via le système
3
Présentation du
code reconstitué
1
Accès au
code
Site Web
XML
pré-transformé
+
XSL
Système
2
Décomposition
contenants/contenus
Système de gestion
de version
Pour les informations
sauvegardées, le système
vérifie si des mises à jour ont
été effectuées depuis la
dernière connexion, et les
propose le cas écheant.

Fig.75

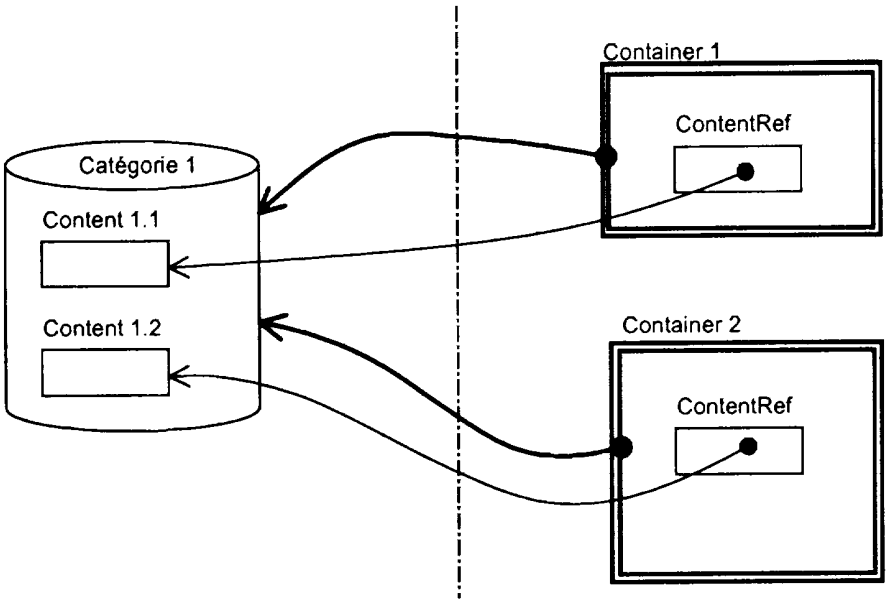

Container 1
ContentRef
Catégorie 1
Content 1.1
Content 1.2
Container 2
ContentRef

Fig.76

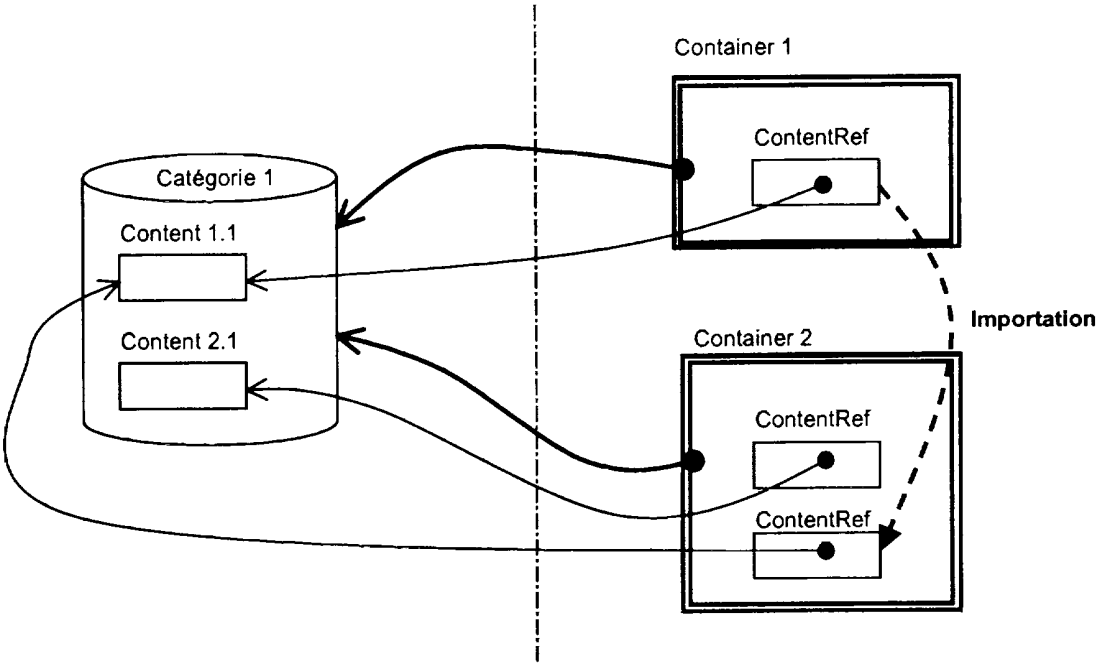

Container 1
ContentRef
Catégorie 1
Content 1.1
Content 2.1
Importation
Container 2
ContentRef
ContentRef

Fig.77

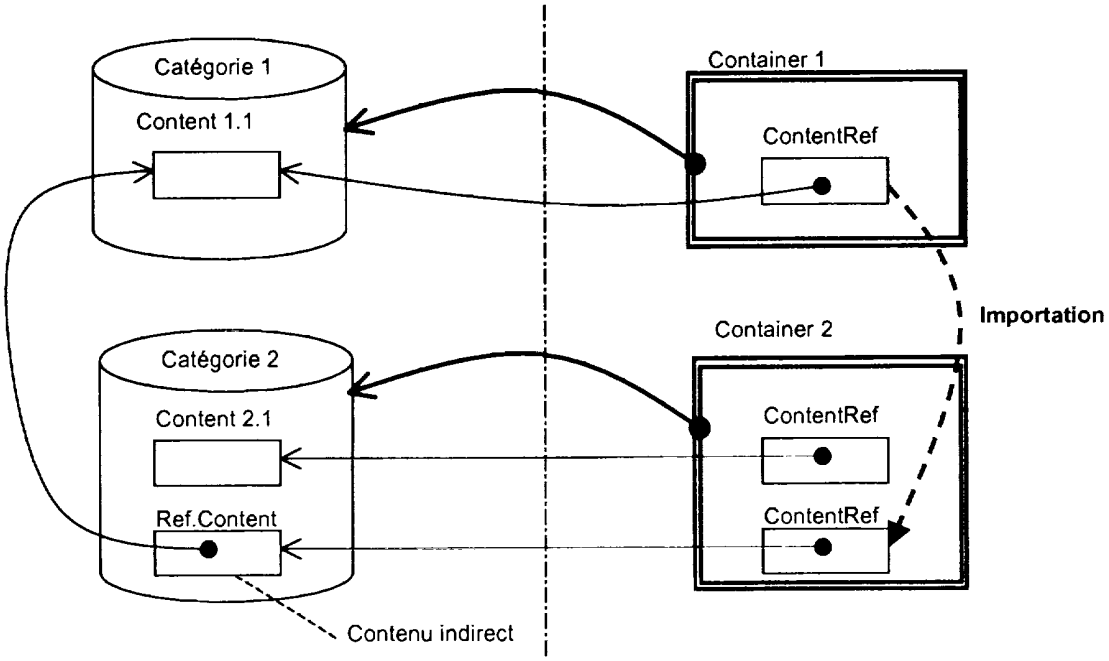

Catégorie 1
Content 1.1
Container 1
ContentRef
Importation
Container 2
Catégorie 2
Content 2.1
ContentRef
Ref.Content
ContentRef
Contenu indirect

Fig.78

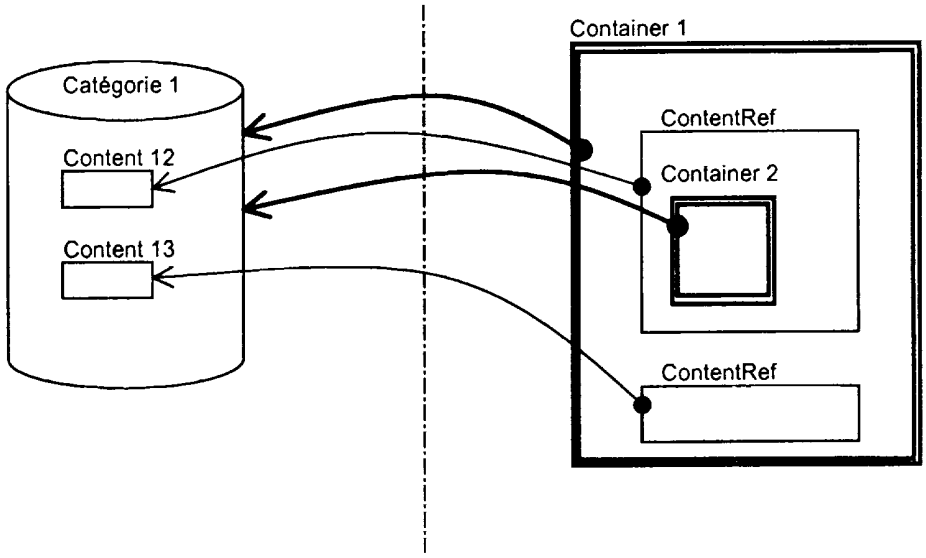


Fig.79

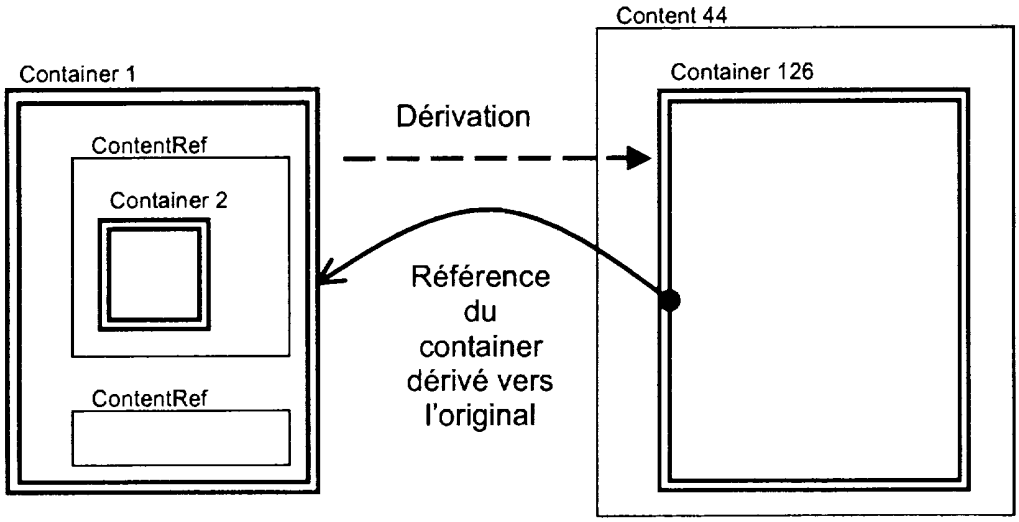


Fig.80a

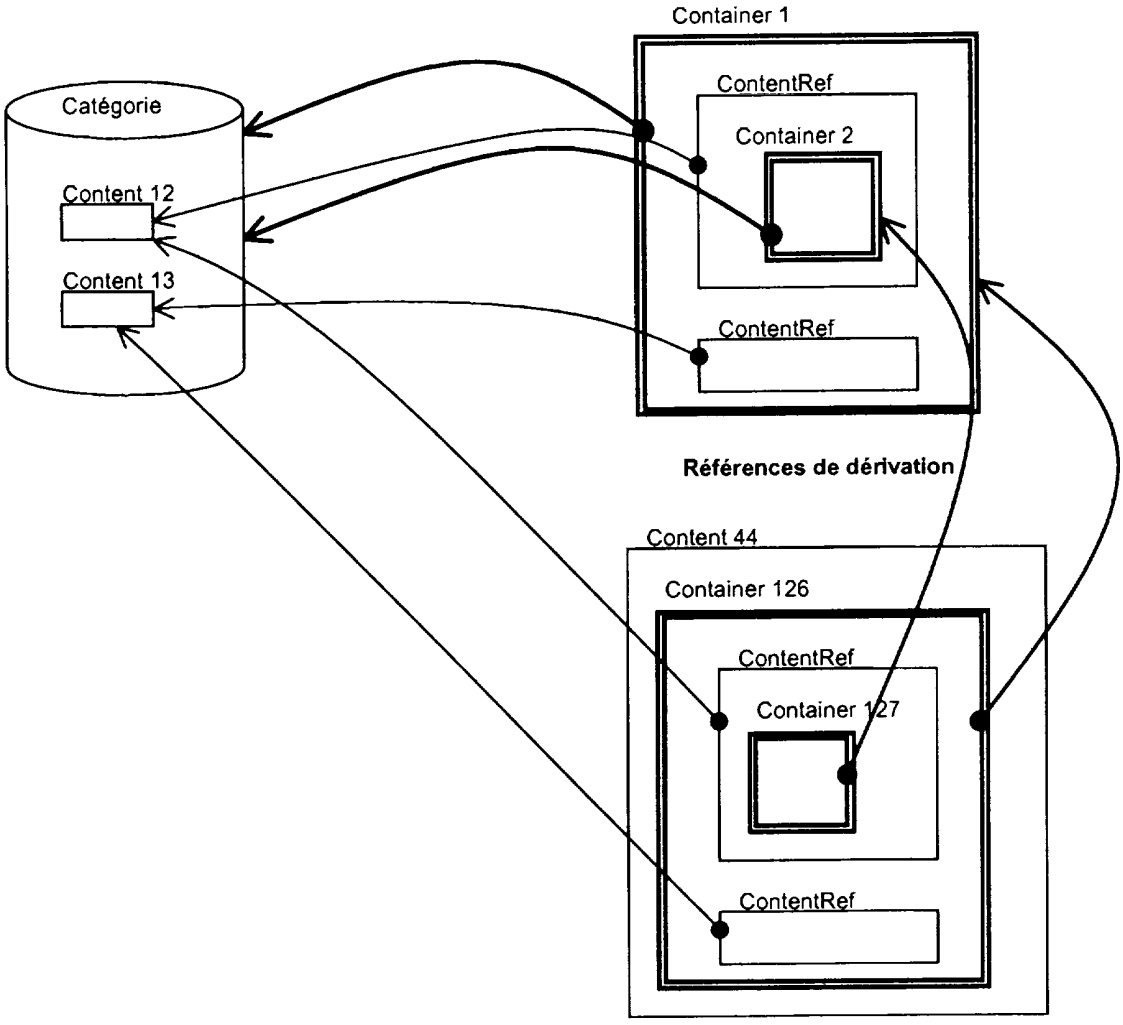


Fig.80b

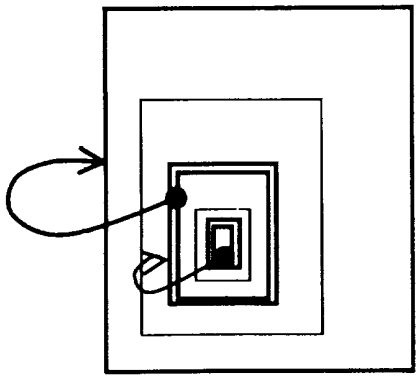


Fig.81

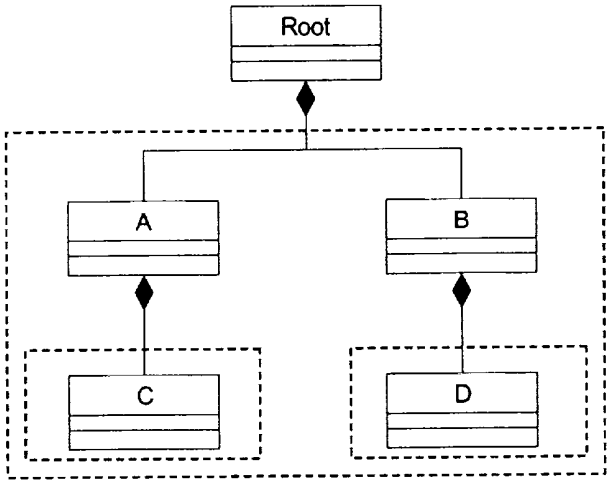

Root
A
B
C
D

Fig.82

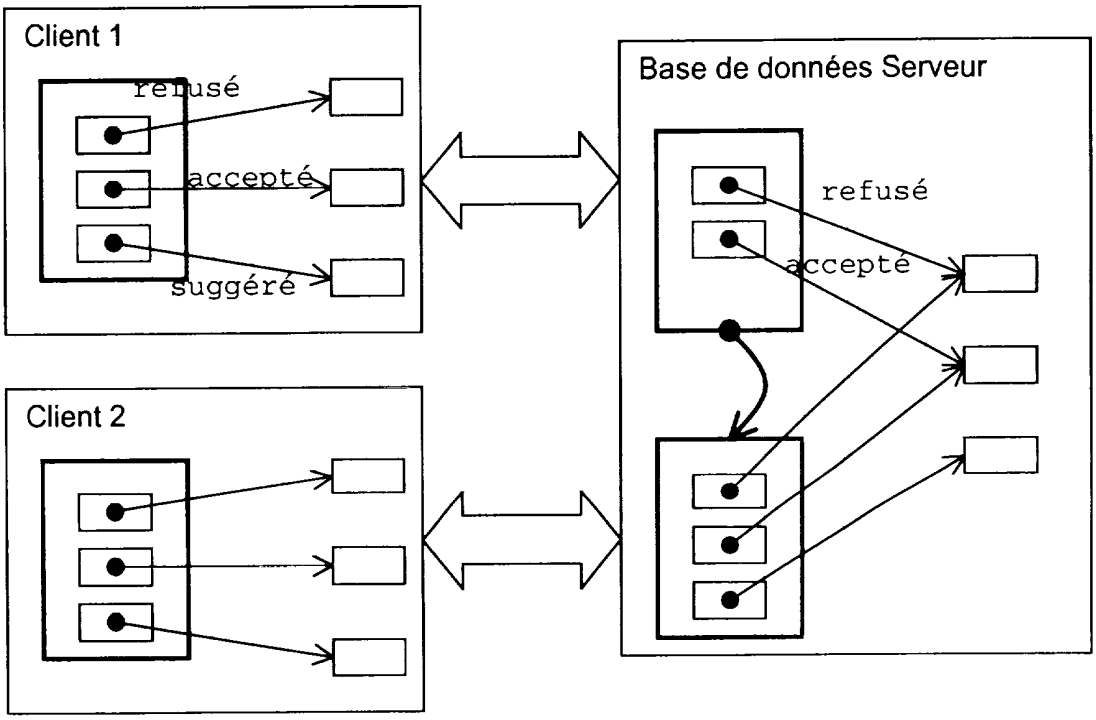

Client 1
refusé
accepté
suggéré
Client 2
Base de données Serveur
refusé
accepté

Fig.83

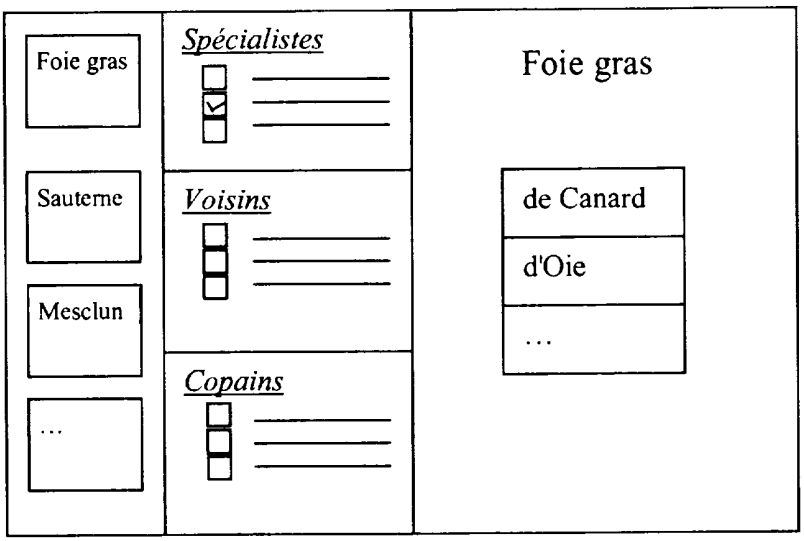


Fig.84a

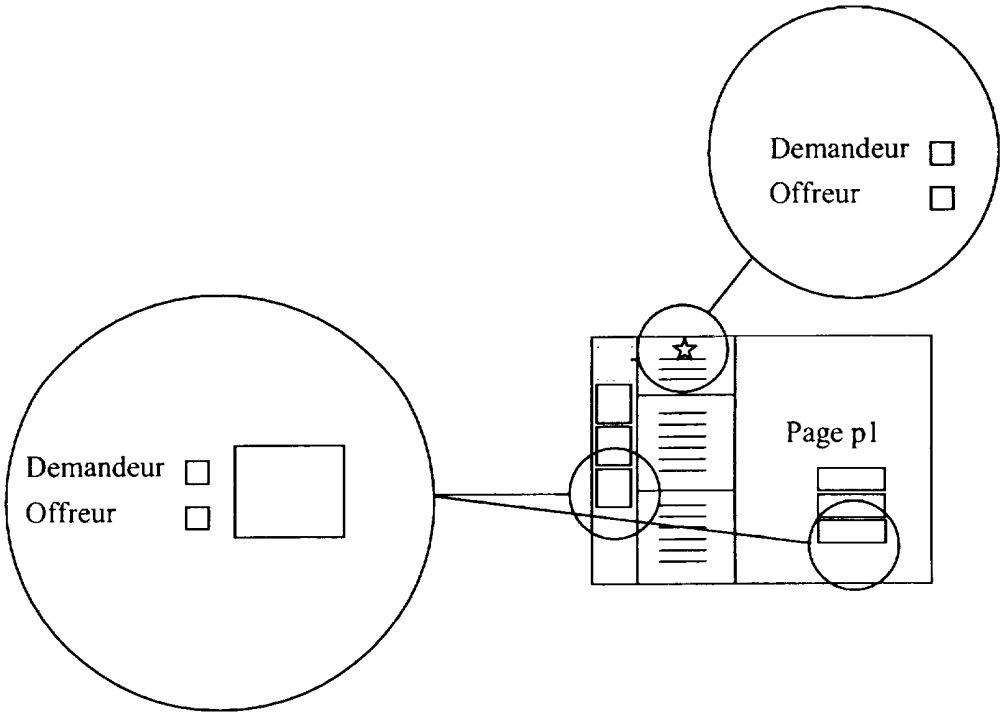


Fig.84b

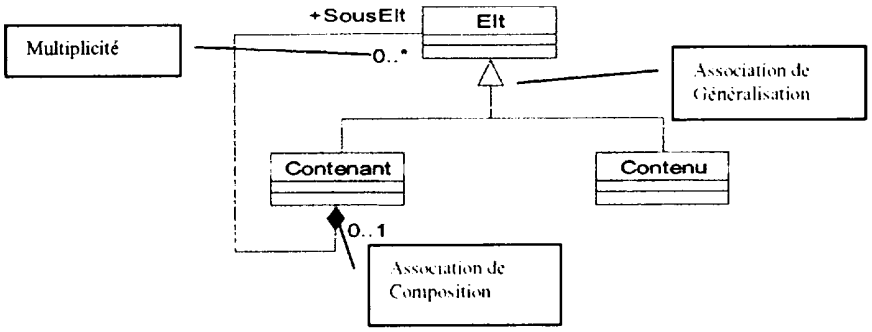

Multiplicité
+SousElt
Elt
0..*
Association de Généralisation
Contenant
Contenu
0..1
Association de Composition

Fig. 85

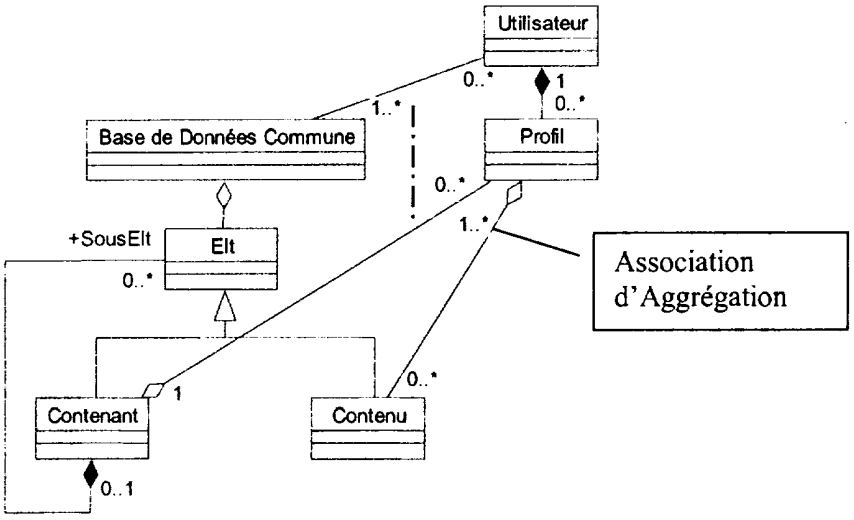

Utilisateur
0..*
1
0..*
1..*
Base de Données Commune
Profil
0..*
1..*
+SousElt
Elt
0..*
Association d'Aggrégation
1
0..*
Contenant
Contenu
0..1

Fig. 86

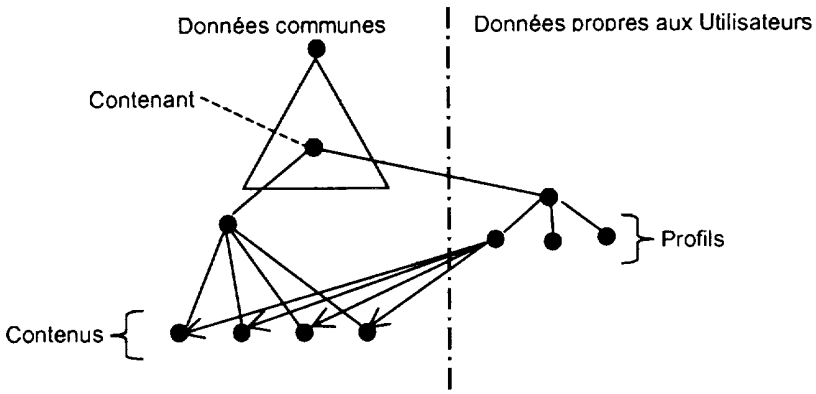

Données communes
Données propres aux Utilisateurs
Contenant
Profils
Contenus

Fig. 87

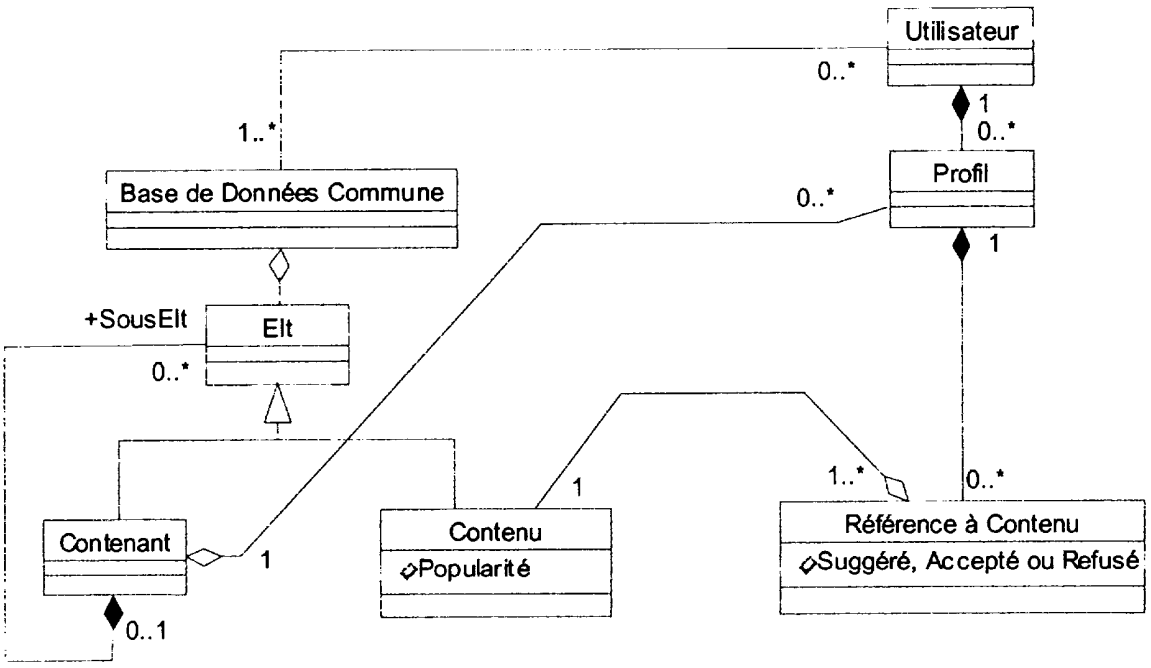

Utilisateur
0..*
1
0..*
1..*
Base de Données Commune
Profil
0..*
1
+SousElt
Elt
0..*
1
1..*
0..*
Contenant
1
Contenu
◇Popularité
Référence à Contenu
◇Suggéré, Accepté ou Refusé
0..1

Fig. 88

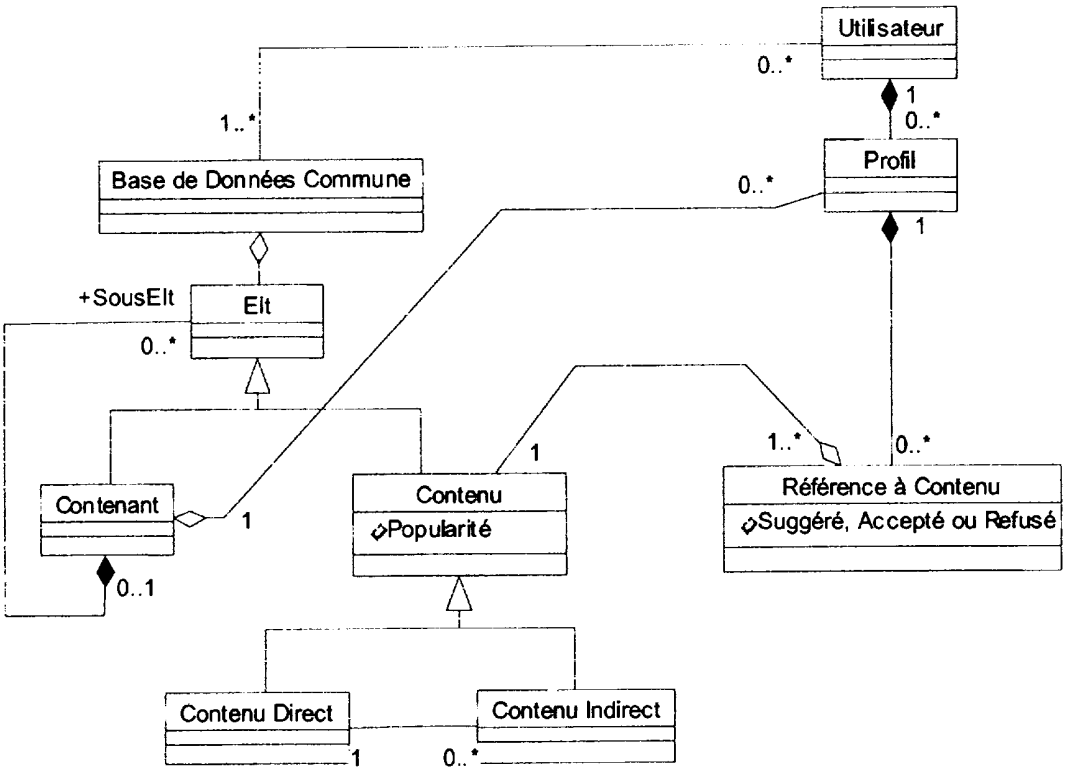

Utilisateur
0..*
1
0..*
1..*
Base de Données Commune
Profil
0..*
1
+SousElt
Elt
0..*
1
1..*
0..*
Contenant
1
Contenu
◇Popularité
Référence à Contenu
◇Suggéré, Accepté ou Refusé
0..1
Contenu Direct
Contenu Indirect
1
0..*

Fig. 89

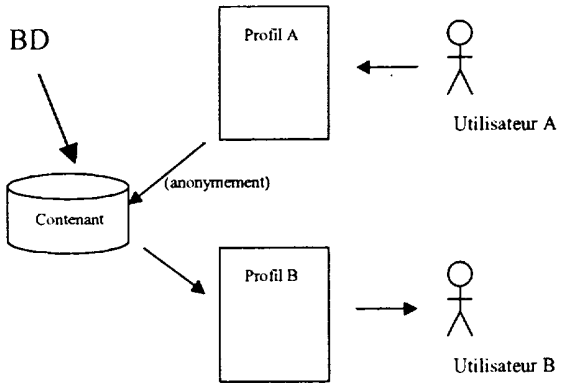

BD
Profil A
Utilisateur A
(anonymement)
Contenant
Profil B
Utilisateur B

Fig. 90

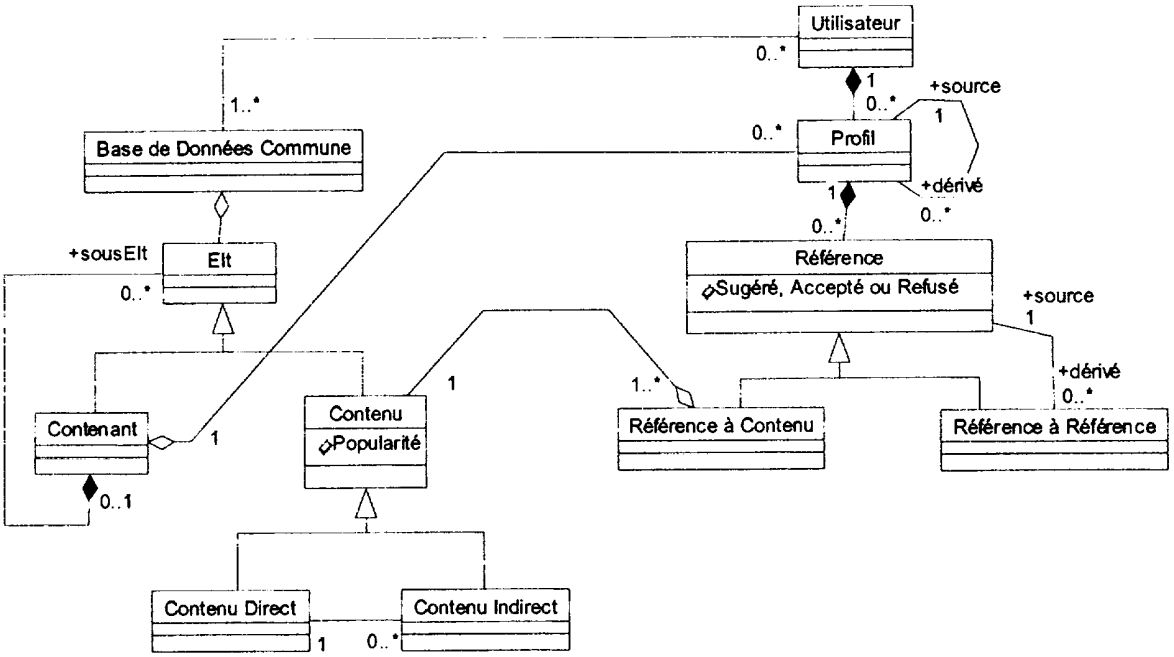


Fig. 91

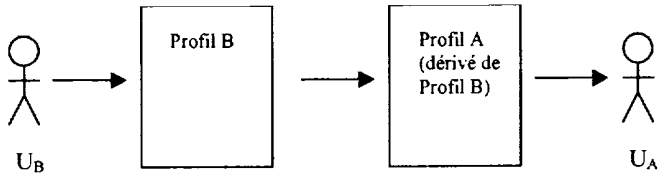


Fig. 92

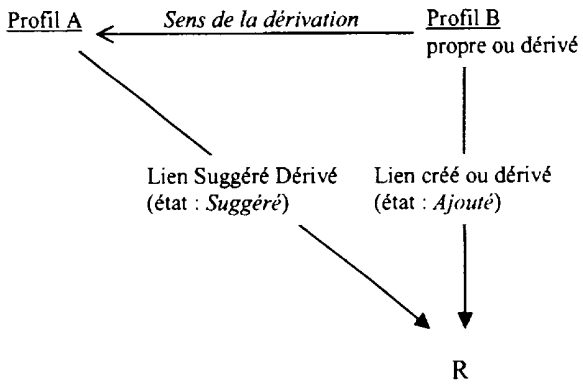


Fig. 93

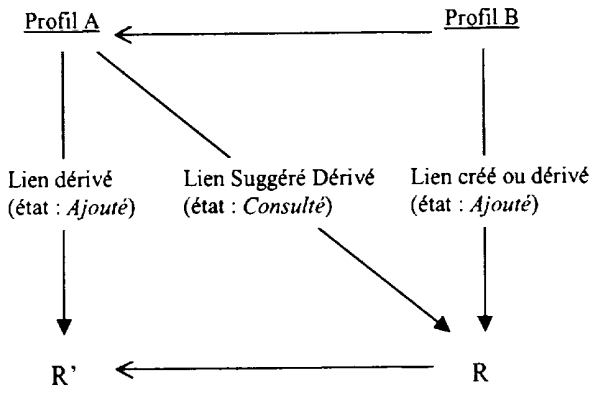


Fig. 94

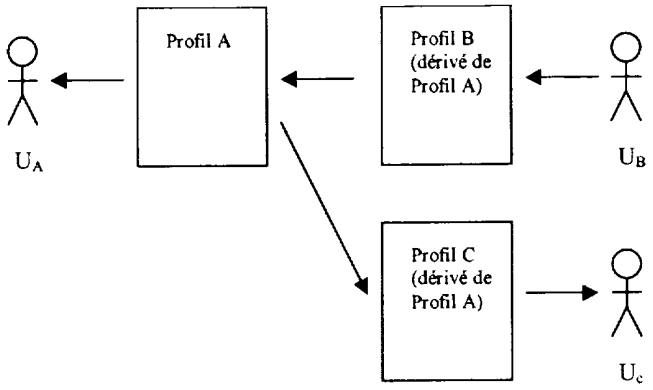


Fig. 95

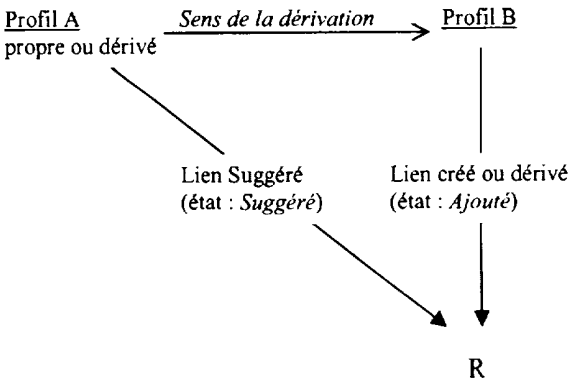


Fig. 96

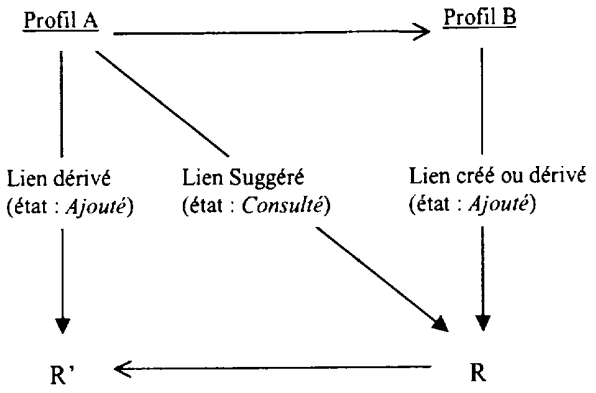


Fig. 97

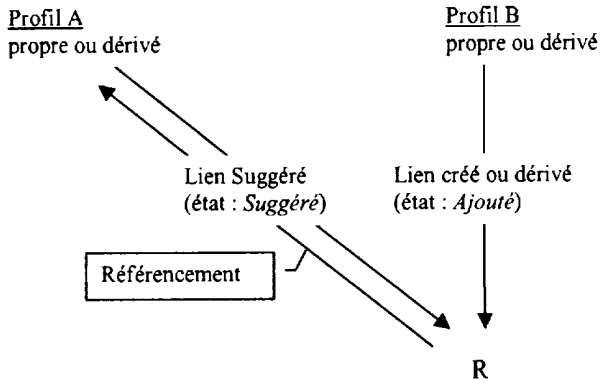


Fig. 98

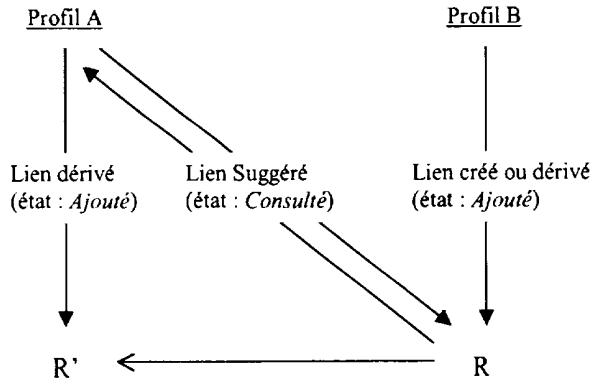


Fig. 99

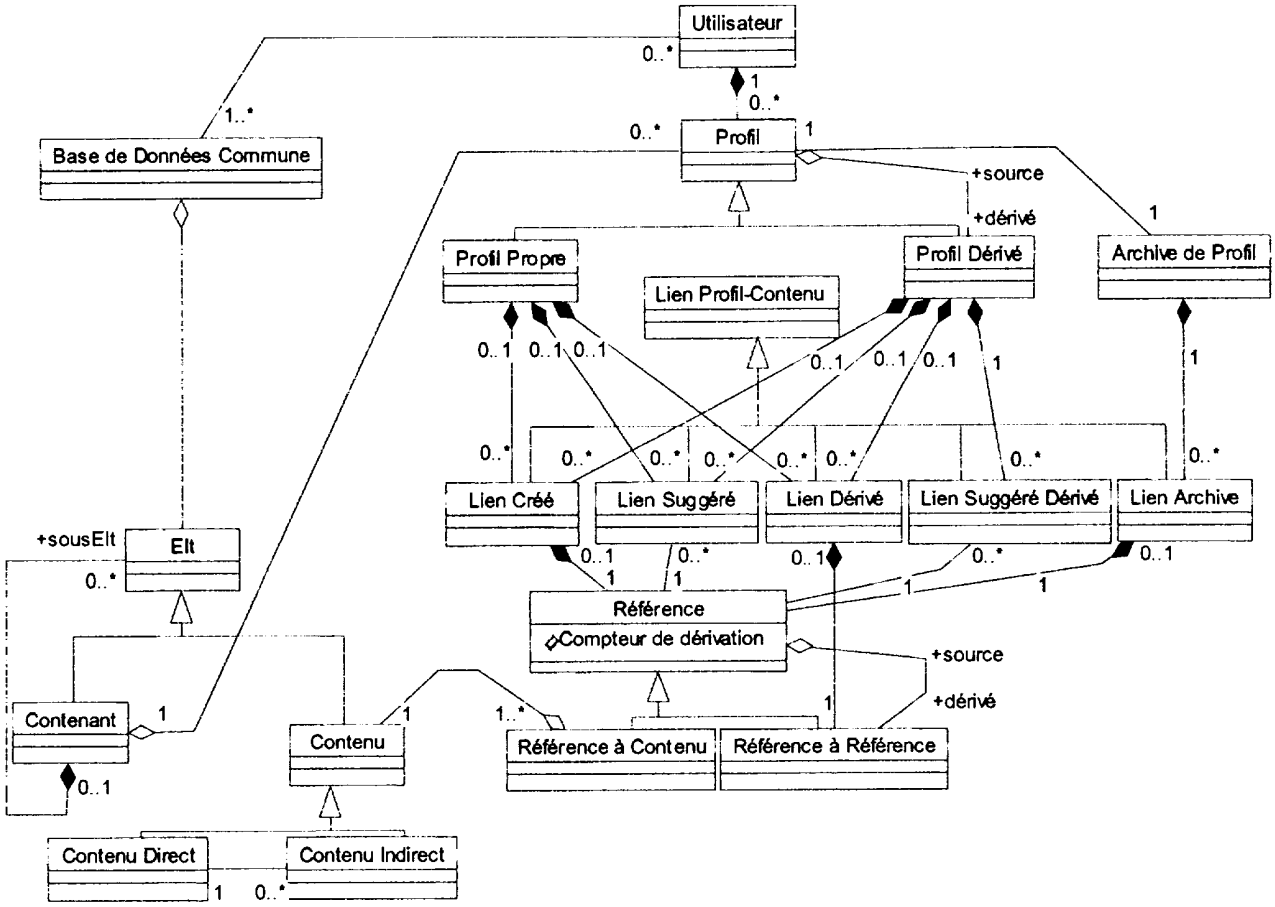


Fig. 100

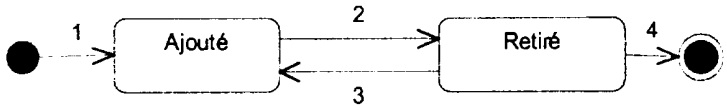


Fig. 101

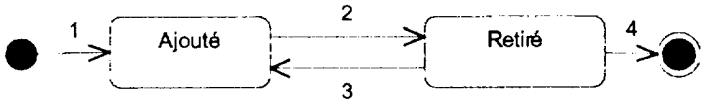


Fig. 102

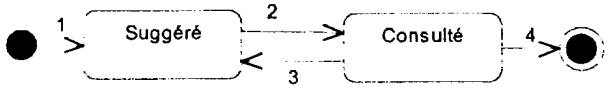


Fig. 103

Fig. 104

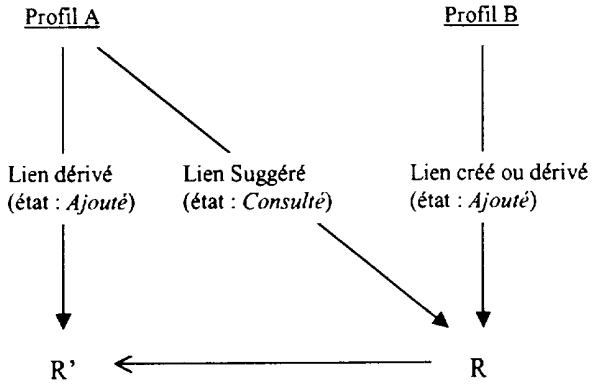


Fig. 105

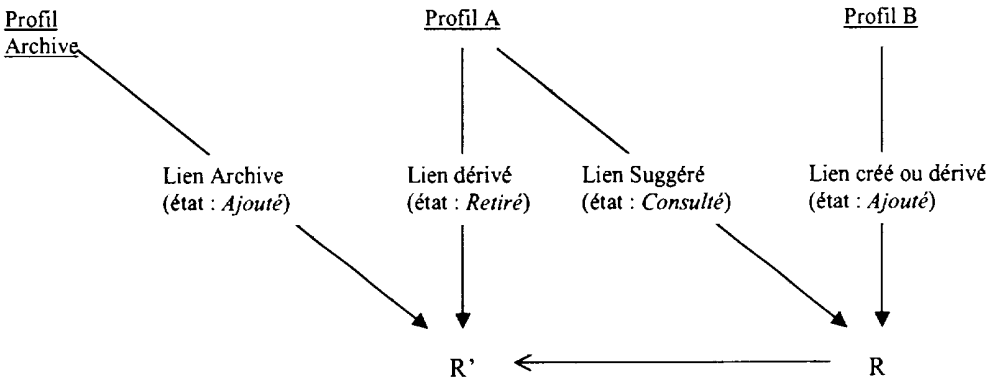


Fig. 106

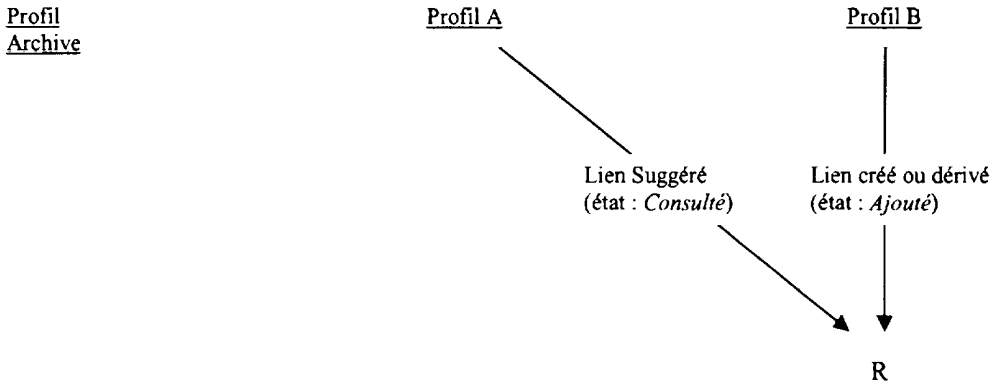


Fig. 107

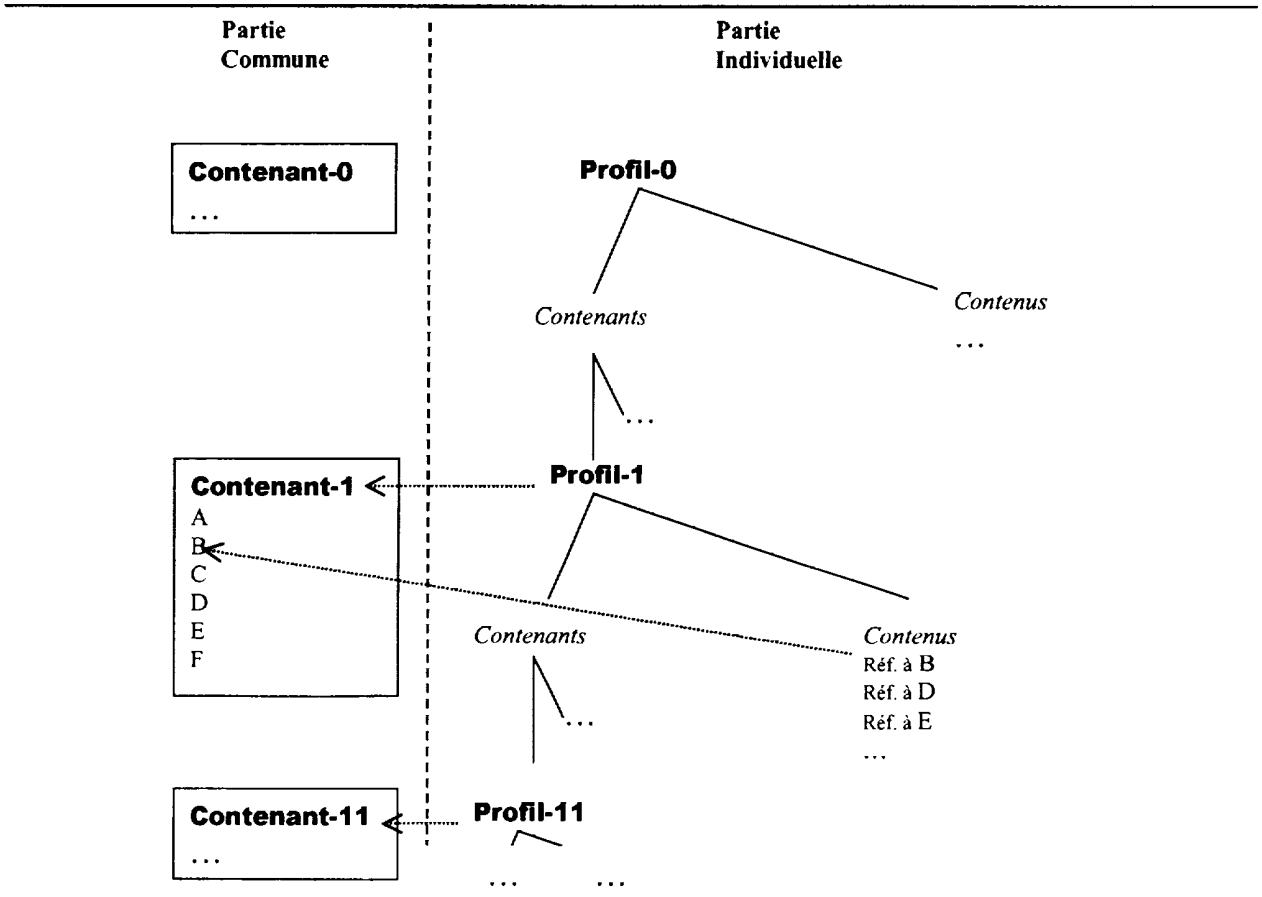


Fig. 108

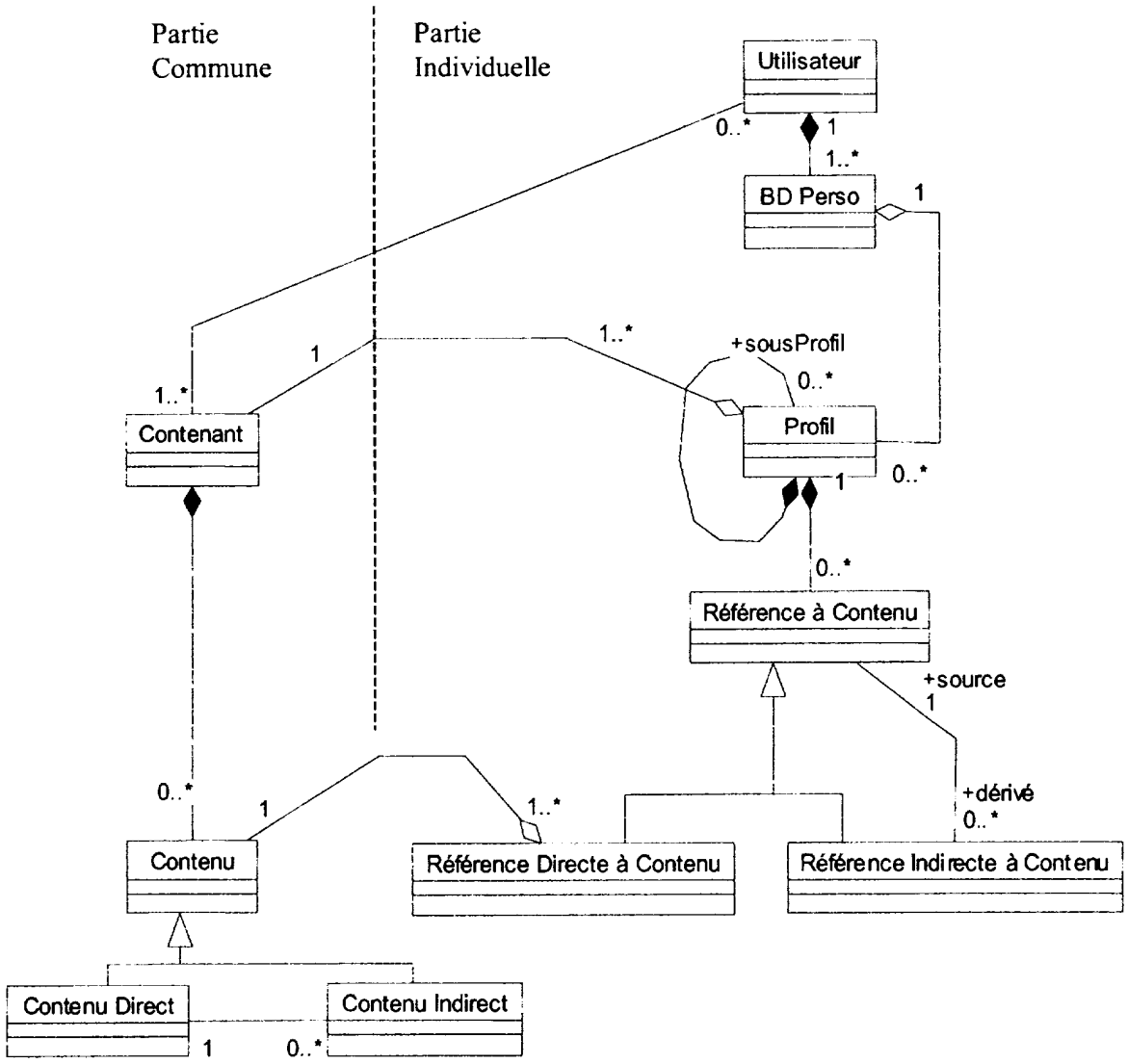

Partie
Commune
Partie
Individuelle
Utilisateur
0..*
1
1..*
BD Perso
1
1..*
+sousProfil
0..*
1
1..*
Contenant
Profil
0..*
1
0..*
Référence à Contenu
+source
1
0..*
1
1..*
+dérivé
0..*
Contenu
Référence Directe à Contenu
Référence Indirecte à Contenu
Contenu Direct
Contenu Indirect
1
0..*

Fig. 109

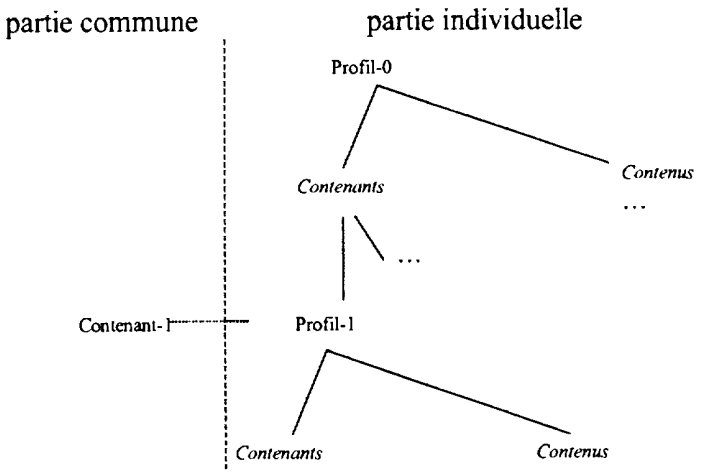

partie commune
partie individuelle
Profil-0
Contenants
Contenus
...
...
Contenant-1
Profil-1
Contenants
Contenus

Fig. 110

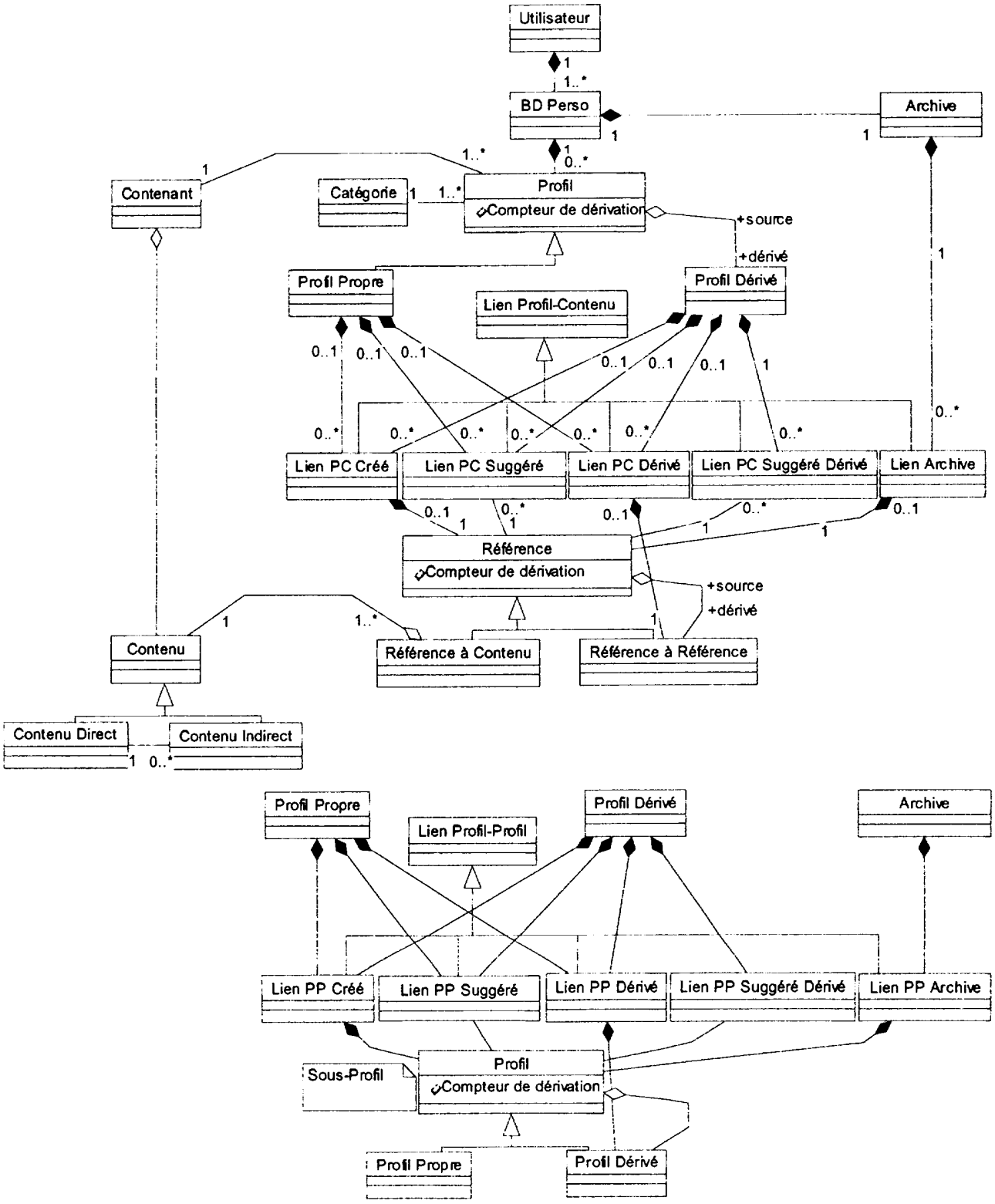

Utilisateur
1
1..*
BD Perso
1
Archive
1
1..*
0..*
Contenant
1
Catégorie
1
1..*
Profil
Compteur de dérivation
+source
+dérivé
1
Profil Propre
Lien Profil-Contenu
Profil Dérivé
0..1
0..1
0..1
0..1
0..1
0..1
1
0..*
0..*
0..*
0..*
0..*
0..*
0..*
Lien PC Créé
Lien PC Suggéré
Lien PC Dérivé
Lien PC Suggéré Dérivé
Lien Archive
0..1
1
0..*
1
0..1
1
0..*
1
0..1
Référence
Compteur de dérivation
+source
+dérivé
1
1..*
1
Contenu
Référence à Contenu
Référence à Référence
Contenu Direct
Contenu Indirect
1
0..*
Profil Propre
Lien Profil-Profil
Profil Dérivé
Archive
Lien PP Créé
Lien PP Suggéré
Lien PP Dérivé
Lien PP Suggéré Dérivé
Lien PP Archive
Sous-Profil
Profil
Compteur de dérivation
Profil Propre
Profil Dérivé

Fig. 111

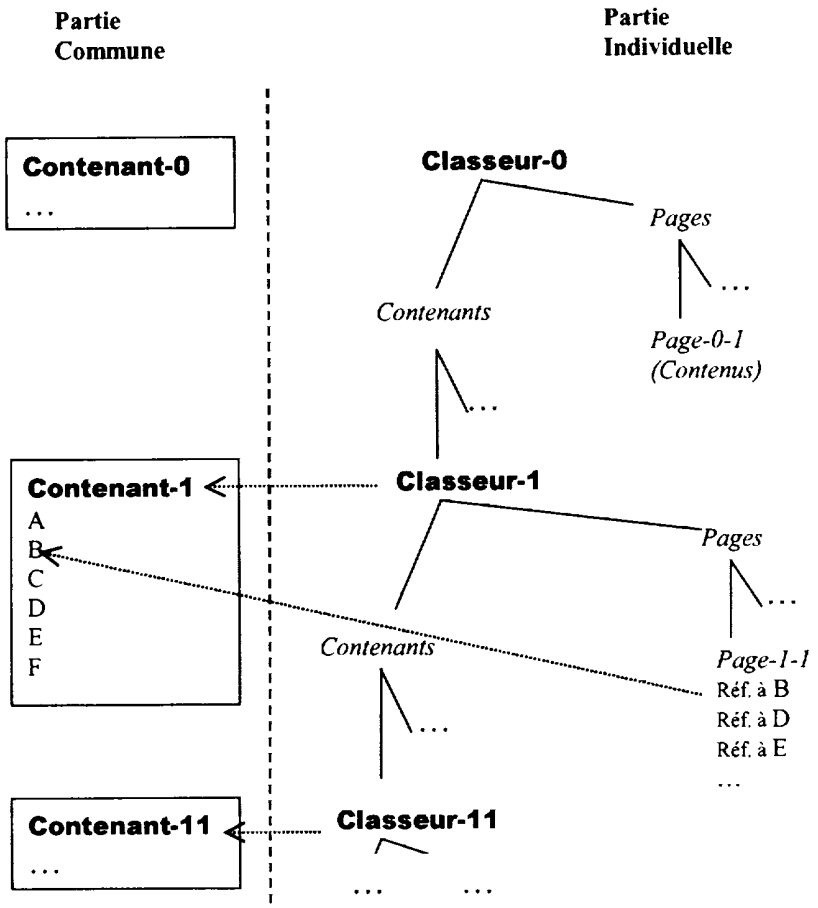

Partie
Commune
Partie
Individuelle
Contenant-0
...
Classeur-0
Pages
...
Contenants
Page-0-1
(Contenus)
...
Contenant-1
A
B
C
D
E
F
Classeur-1
Pages
...
Contenants
Page-1-1
Réf. à B
Réf. à D
Réf. à E
...
...
Contenant-11
...
Classeur-11
... ...

Fig. 112

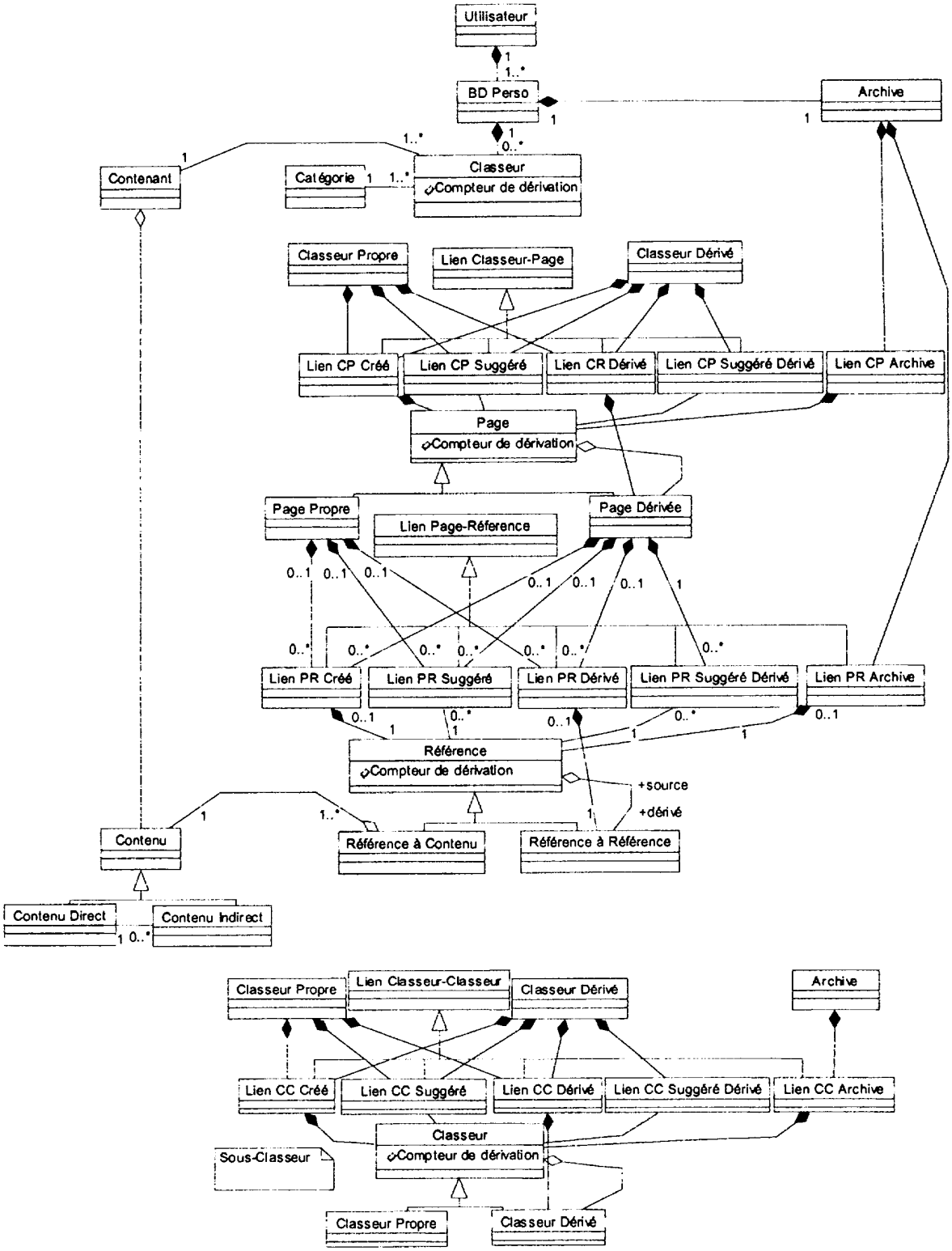

Utilisateur
BD Perso
Archive
Contenant
Catégorie
Classeur
Compteur de dérivation
Classeur Propre
Lien Classeur-Page
Classeur Dérivé
Lien CP Créé
Lien CP Suggéré
Lien CR Dérivé
Lien CP Suggéré Dérivé
Lien CP Archive
Page
Compteur de dérivation
Page Propre
Lien Page-Référence
Page Dérivée
Lien PR Créé
Lien PR Suggéré
Lien PR Dérivé
Lien PR Suggéré Dérivé
Lien PR Archive
Référence
Compteur de dérivation
+source
+dérivé
Contenu
Référence à Contenu
Référence à Référence
Contenu Direct
Contenu Indirect
Classeur Propre
Lien Classeur-Classeur
Classeur Dérivé
Archive
Lien CC Créé
Lien CC Suggéré
Lien CC Dérivé
Lien CC Suggéré Dérivé
Lien CC Archive
Classeur
Compteur de dérivation
Sous-Classeur
Classeur Propre
Classeur Dérivé

Fig. 113

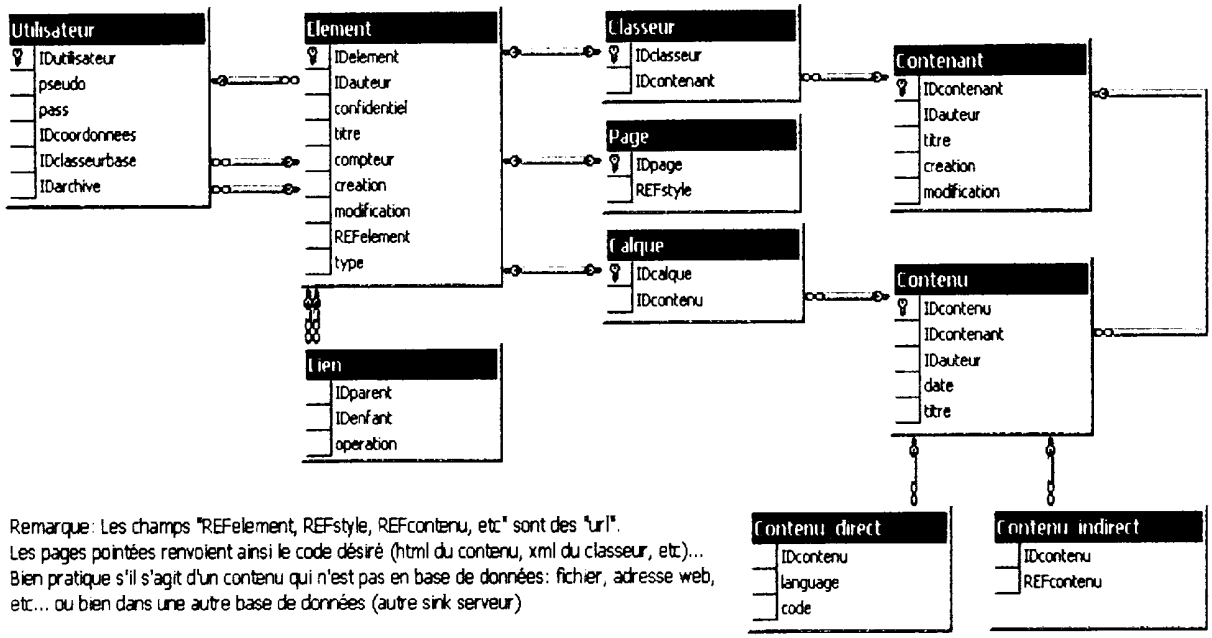
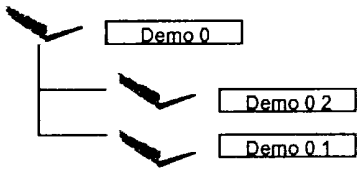


Fig. 114

Fig. 115

IDutilisateur	Pseudo	IDclasseurbase
1	Demo	1
2	Toto	8
3	Seb	9
4	Aline	10

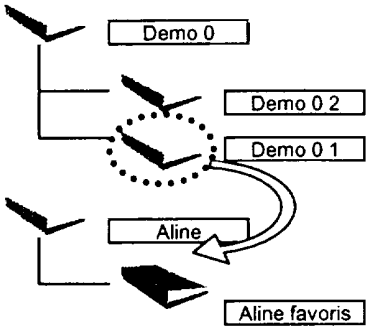


IDelement	IDauteur	Compteur	Type	REFelement	Titre
1	2	0	1	Null	demo 0
2	2	0	1	Null	demo 0 1
3	2	0	2	Null	demo page 1
4	2	0	3	Null	demo calque 1
5	2	0	3	Null	demo calque 2
6	2	0	1	Null	demo 0 2
7	2	0	2	Null	demo page 2

Fig. 116

IDparent	IDenfant	Operation
1	2	+
1	6	+

Fig. 117



IDelement	IDauteur	Compteur	Type	REFelement	Titre
1	2	0	1	Null	demo 0
2	2	1	1	Null	demo 0 1
8	4	0	1	Null	Aline
9	4	0	1	2	Aline favoris

Fig. 118

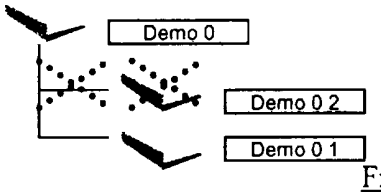


Fig. 119

IDparent	IDenfant	Operation
1	2	+
1	6	+

IDelement	IDauteur	Compteur	Type	REFelement	Titre
1	2	0	1	Null	demo 0
2	2	1	1	Null	demo 0 1
6	2	0	1	Null	demo 0 2

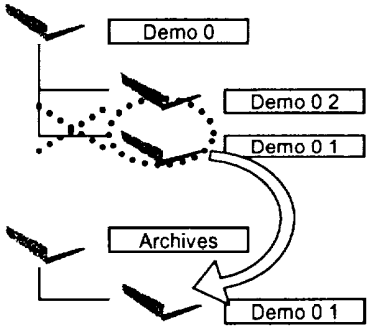


Fig. 120a

IDelement	IDauteur	Compteur	Type	REFelement	Titre
1	2	0	1	Null	demo 0
2	2	1	1	Null	demo 0 1
6	2	0	1	Null	demo 0 2
12	2	0	0	Null	Archive

Fig. 120b

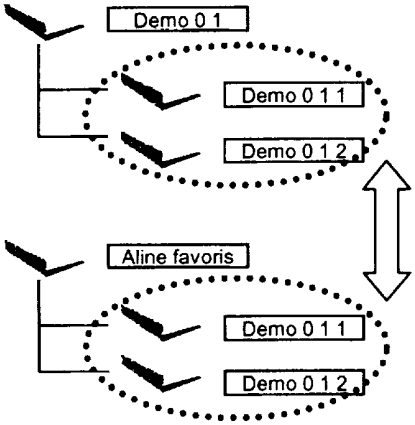
Fig.120c

IDparent	IDenfant	Operation
1	2	0
12	2	+
1	6	+

Fig. 121

IDparent	IDenfant	Operation
1	2	0
12	2	+
1	6	+
IDparent	IDenfant	Operation
1	2	+
1	6	+

Fig. 122



IDelement	IDauteur	Compteur	Type	REFelement	Titre
2	2	1	1	Null	Demo 0 1
9	4	0	1	2	Aline favoris
13	2	0	1	Null	Demo 0 1 1
14	2	0	1	Null	Demo 0 1 2

Fig.123

IDparent	IDenfant	Operation
2	13	+
2	14	+

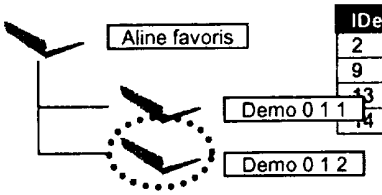
IDelement	IDauteur	Compteur	Type	REFelement	Titre
2	2	1	1	Null	Demo 0 1
9	4	0	1	2	Aline favoris
13	2	0	1	Null	Demo 0 1 1
14	2	0	1	Null	Demo 0 1 2
15	4	0	1	13	Demo 0 1 1

Fig. 124

IDparent	Idenfant	Operation
9	15	+

Fig. 125

Fig. 126



IDelement	IDauteur	Compteur	Type	REFelement	Titre
2	2	1	1	Null	Demo 0 1
9	4	0	1	2	Aline favoris
13	2	0	1	Null	Demo 0 1 1
14	2	0	1	Null	Demo 0 1 2

IDparent	IDenfant	Operation
9	14	-

Fig. 127

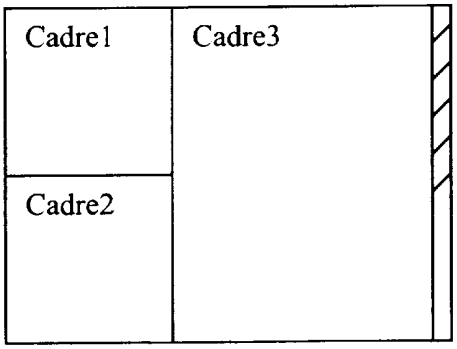


Fig. 127a

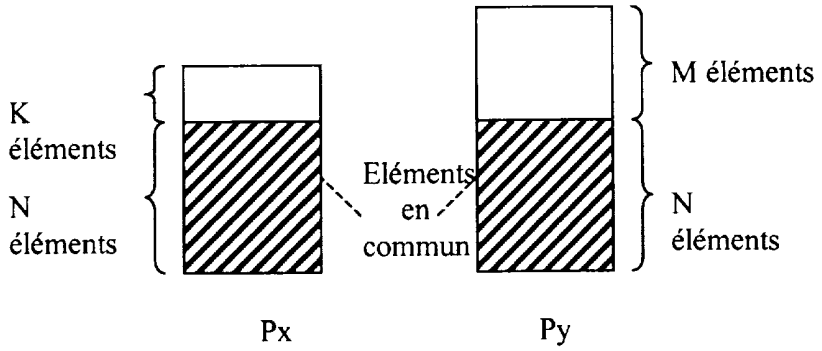


Fig. 128

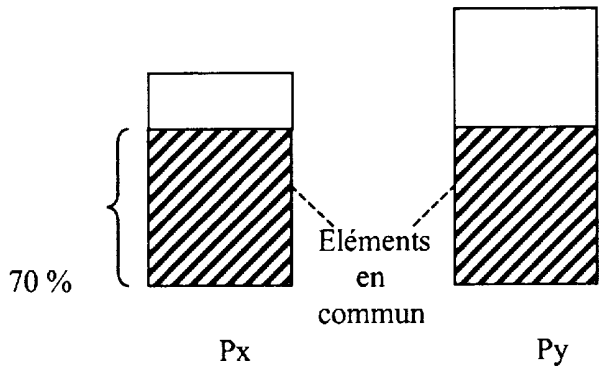


Fig. 129

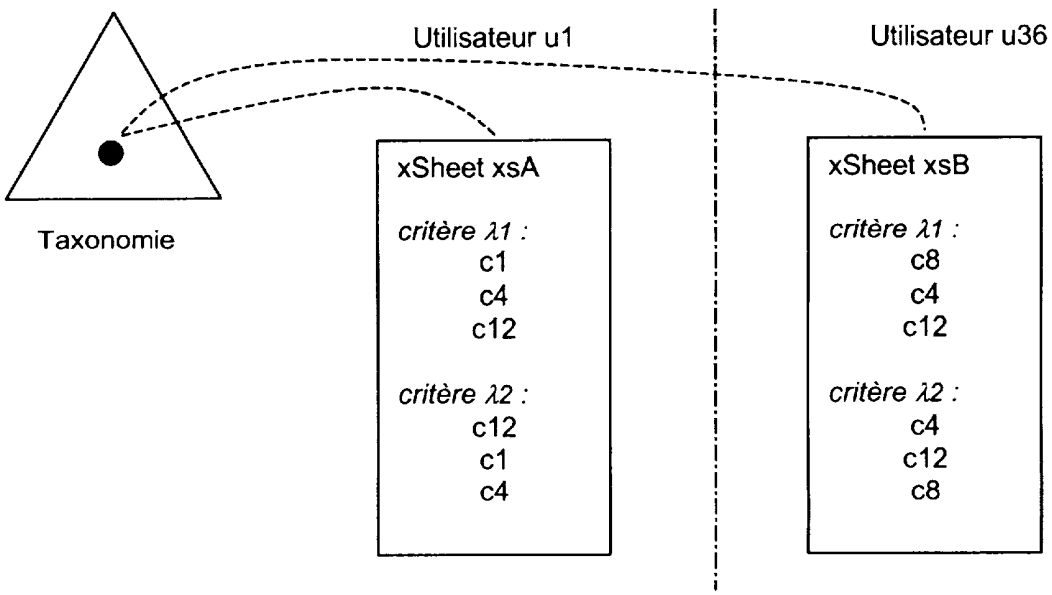


Fig. 130

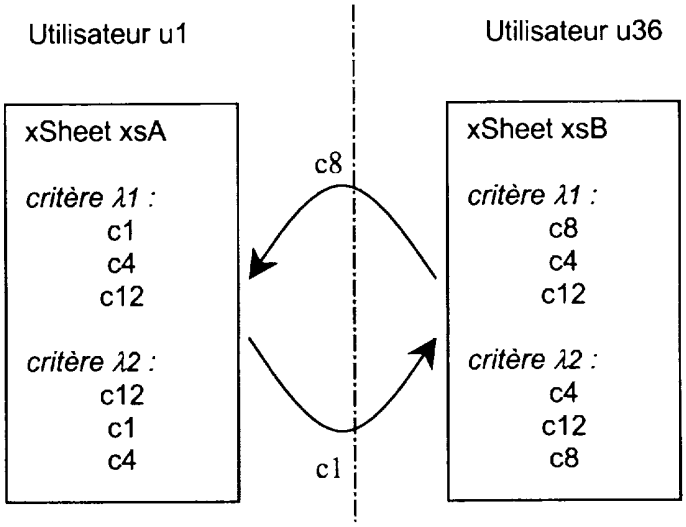


Fig. 131

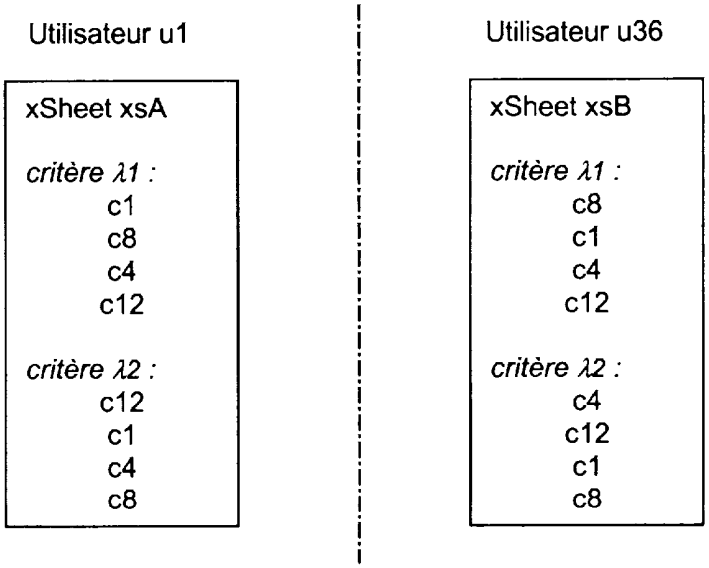


Fig. 132

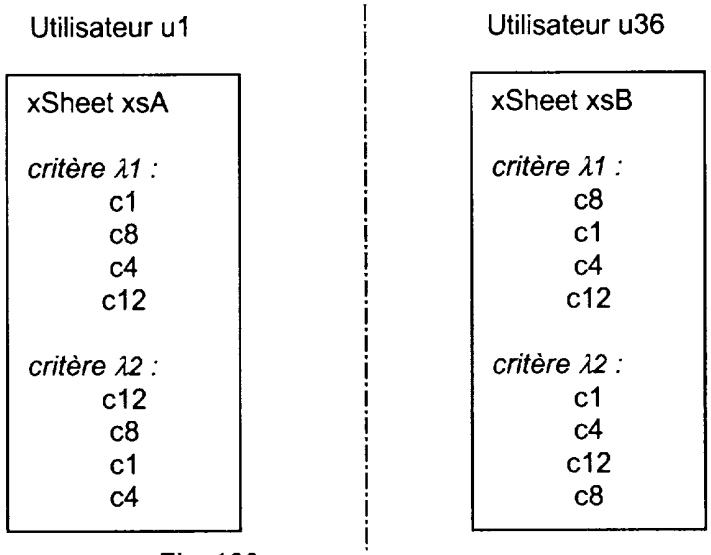


Fig. 133

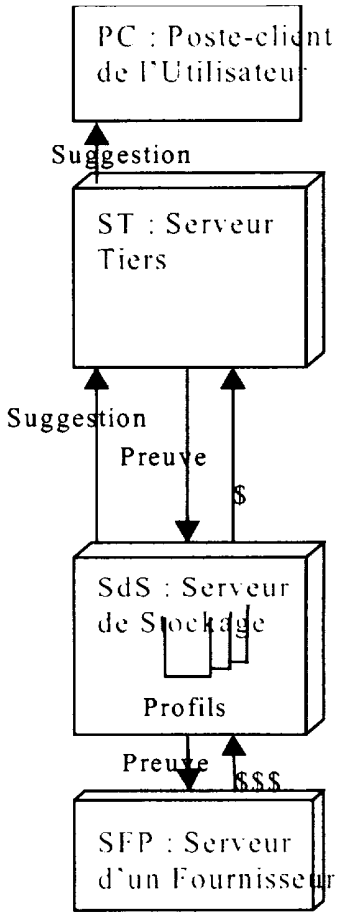


Fig. 134

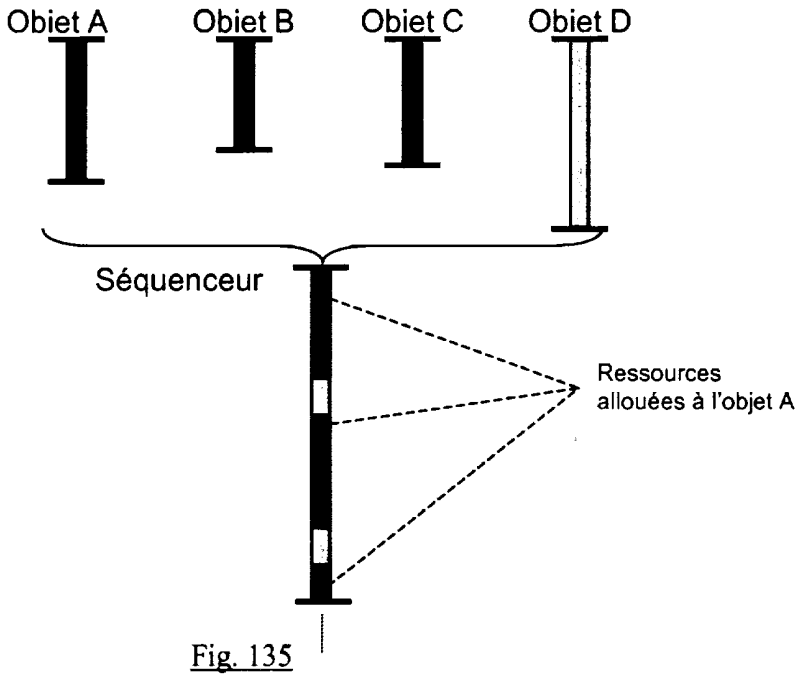


Fig. 135

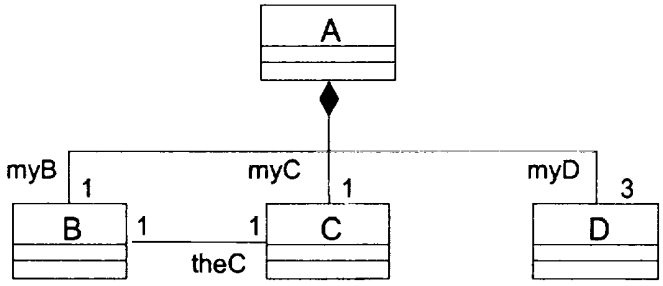


Fig. 136

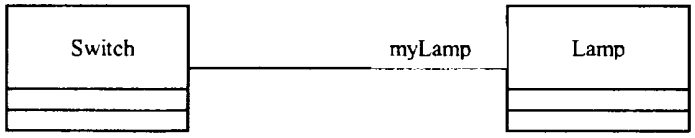


Fig. 137

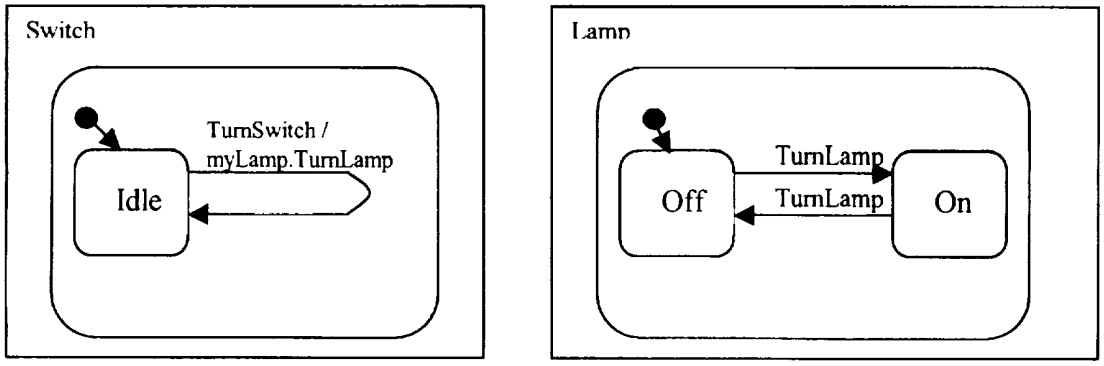


Fig. 138

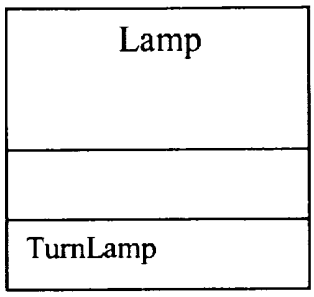


Fig. 139

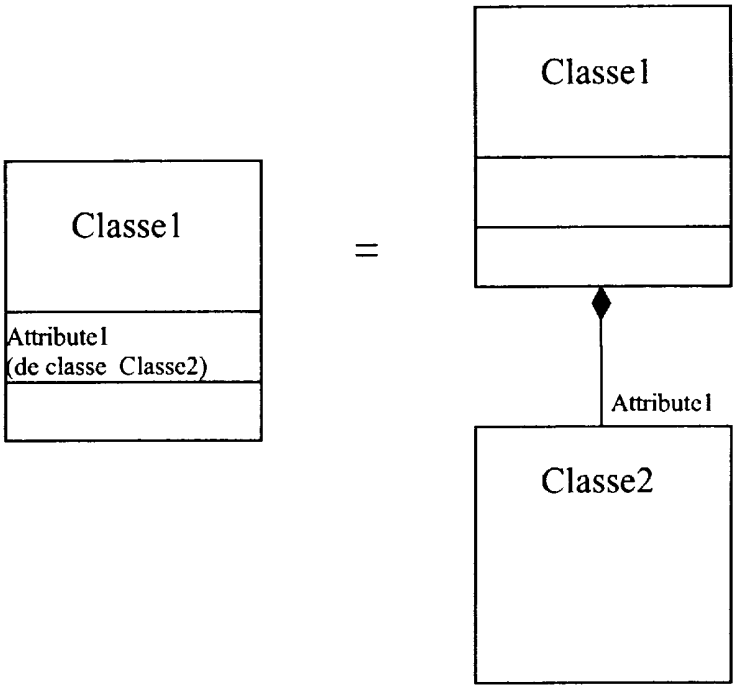


Fig. 140

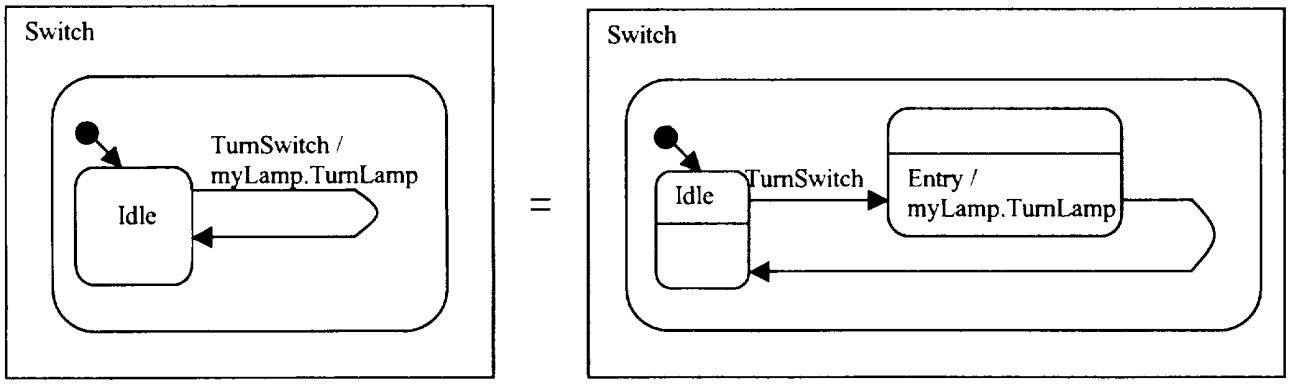


Fig. 141

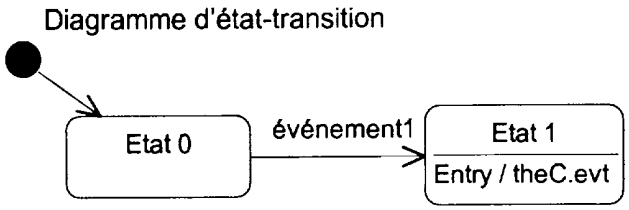


Fig. 142

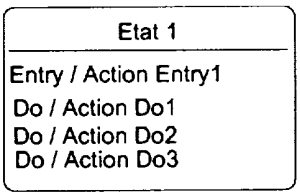


Fig. 143

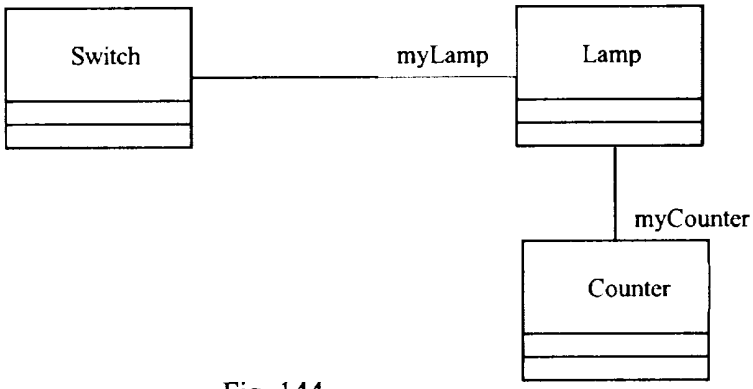


Fig. 144

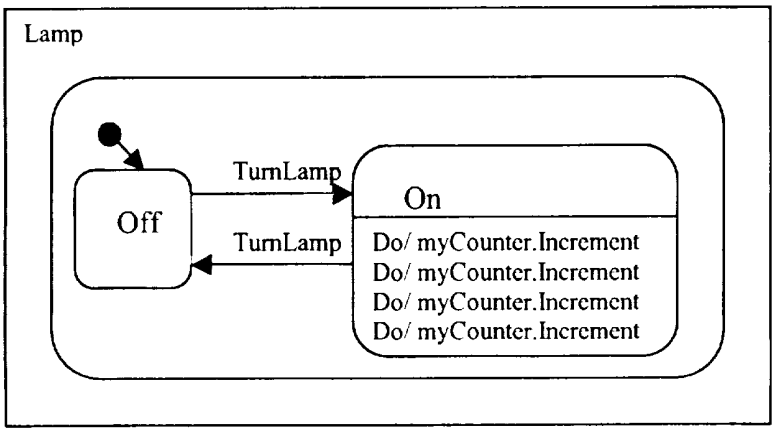


Fig. 145

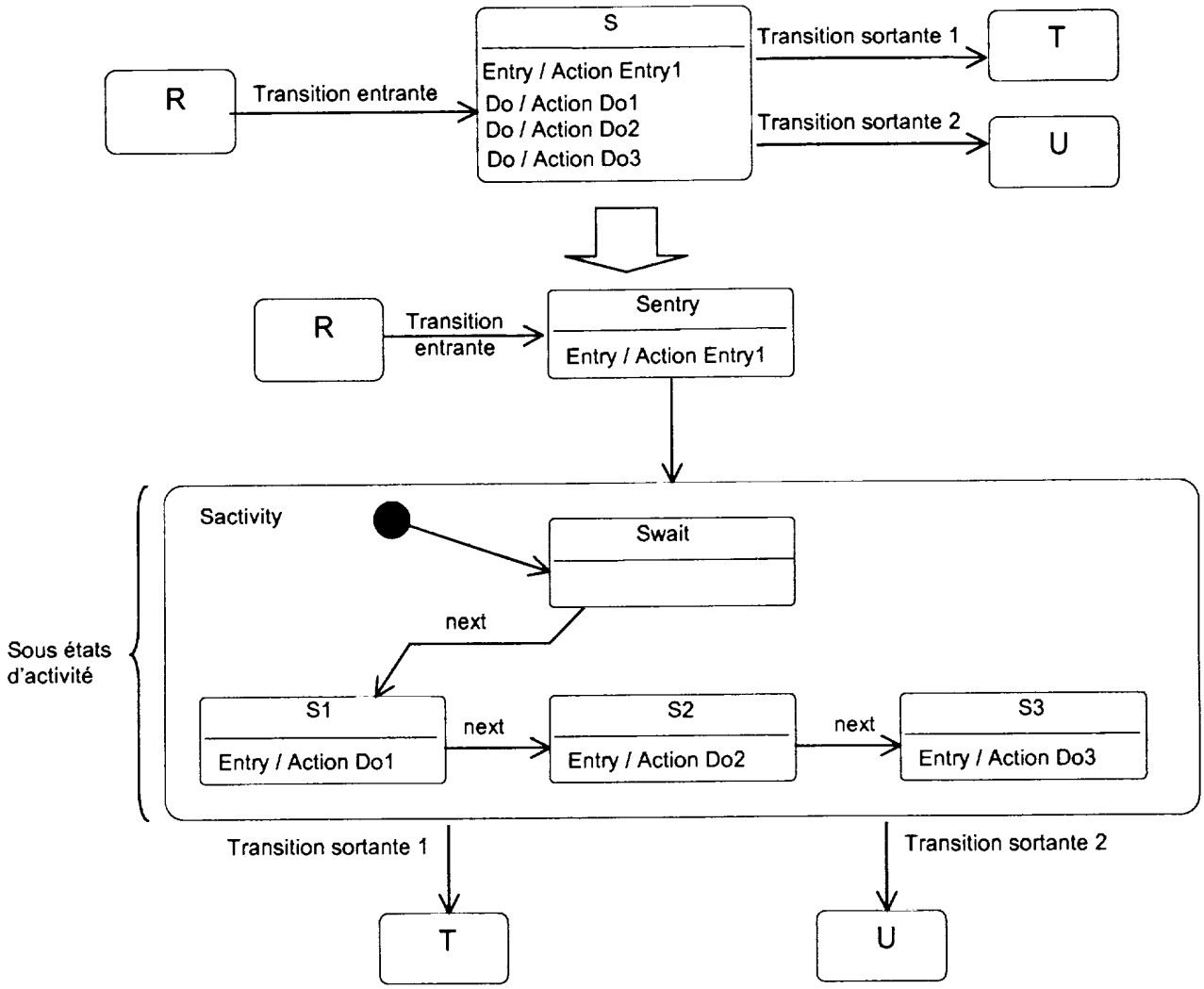

S
Entry / Action Entry1
Do / Action Do1
Do / Action Do2
Do / Action Do3
R
Transition entrante
Transition sortante 1
Transition sortante 2
T
U
Sentry
Entry / Action Entry1
Transition entrante
Sactivity
Swait
next
Sous états d'activité
S1
Entry / Action Do1
S2
Entry / Action Do2
S3
Entry / Action Do3
Transition sortante 1
Transition sortante 2

Fig. 146

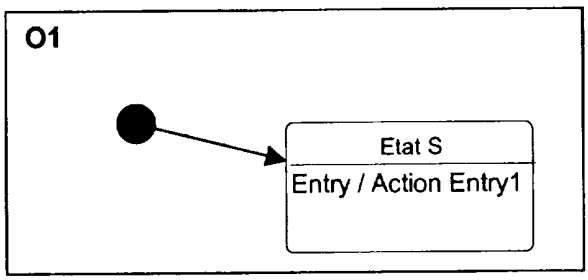


Fig. 147

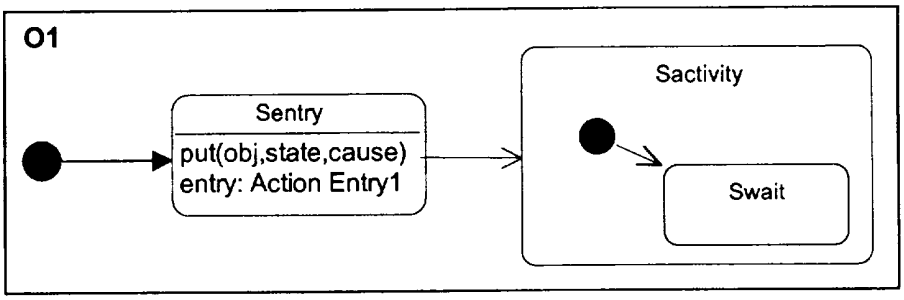


Fig. 148

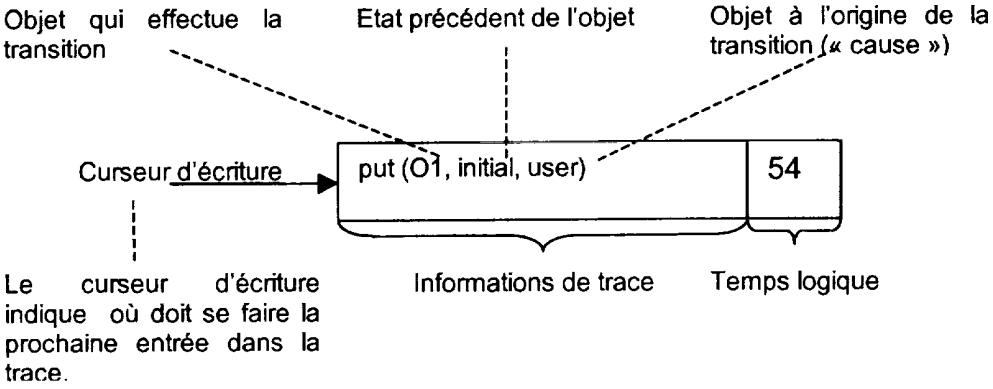


Fig. 149

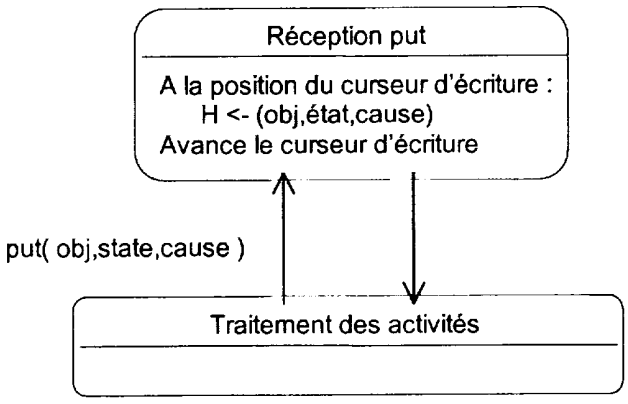


Fig. 150

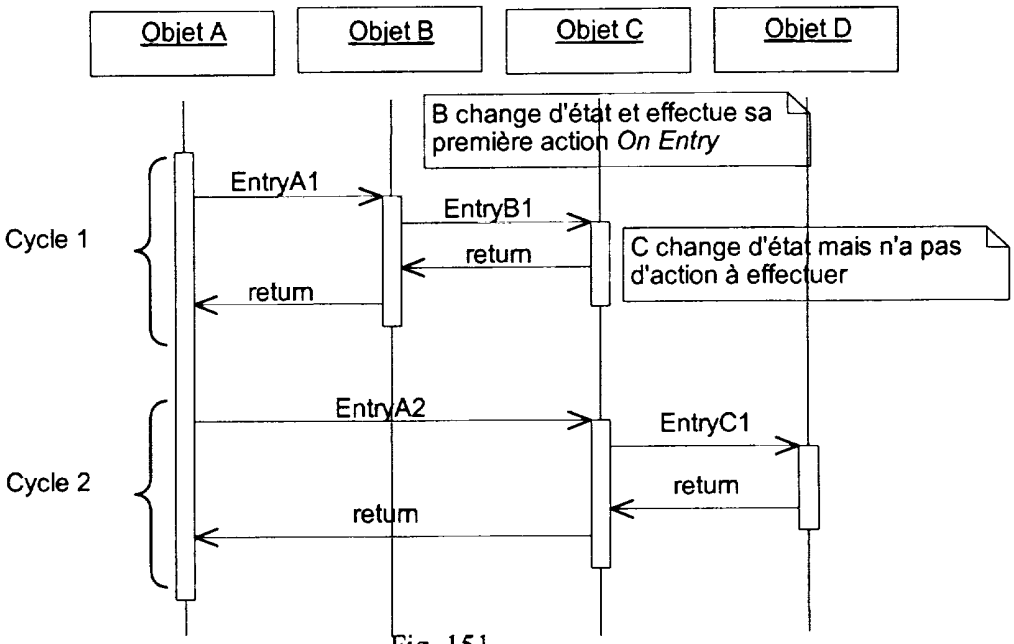

Objet A
Objet B
Objet C
Objet D
B change d'état et effectue sa première action *On Entry*
EntryA1
EntryB1
return
C change d'état mais n'a pas d'action à effectuer
Cycle 1
return
EntryA2
EntryC1
Cycle 2
return
return

Fig. 151

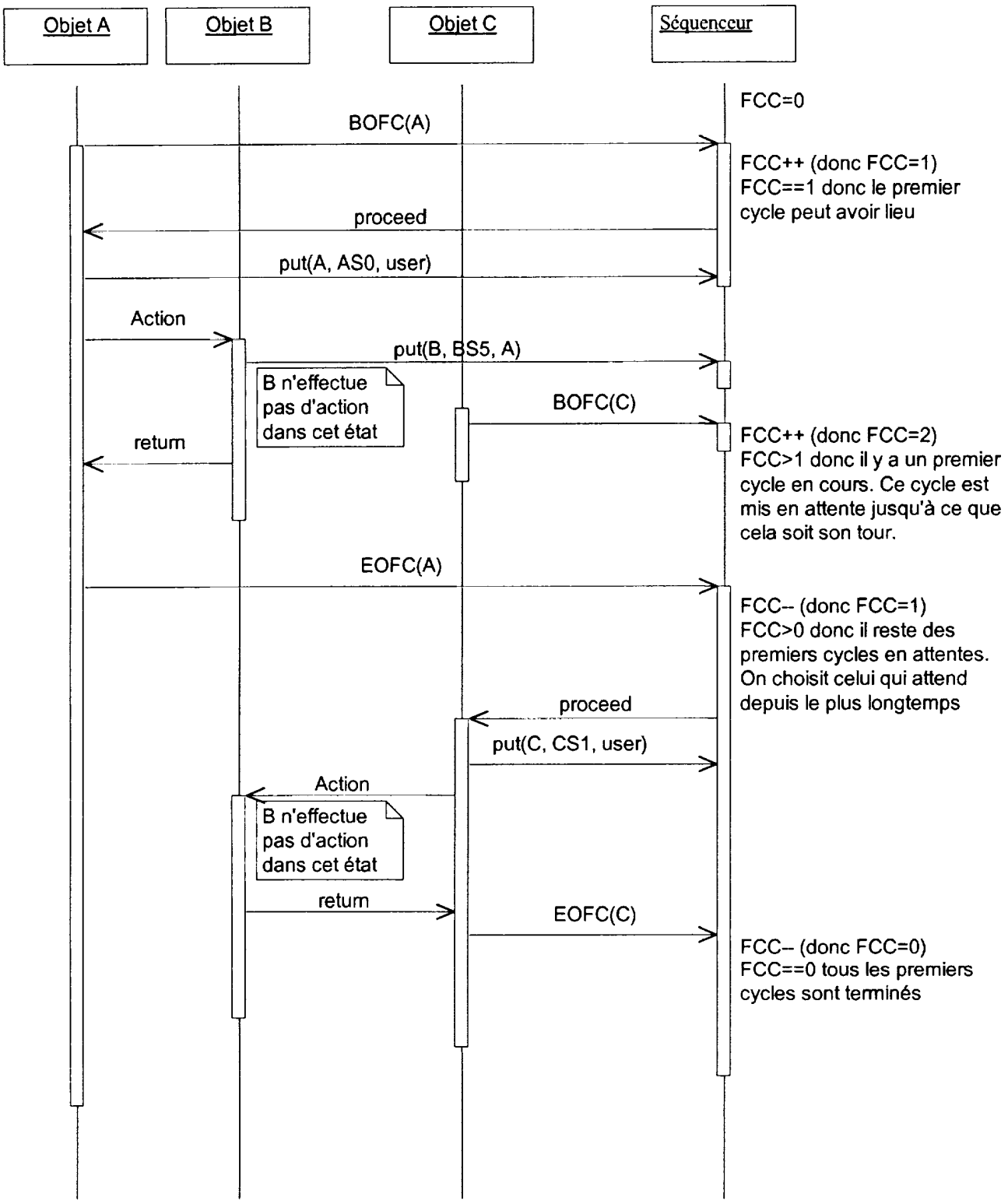

Objet A
Objet B
Objet C
Séquenceur
FCC=0
BOFC(A)
FCC++ (donc FCC=1)
FCC==1 donc le premier cycle peut avoir lieu
proceed
put(A, AS0, user)
Action
put(B, BS5, A)
B n'effectue pas d'action dans cet état
BOFC(C)
return
FCC++ (donc FCC=2)
FCC>1 donc il y a un premier cycle en cours. Ce cycle est mis en attente jusqu'à ce que cela soit son tour.
EOFC(A)
FCC-- (donc FCC=1)
FCC>0 donc il reste des premiers cycles en attentes. On choisit celui qui attend depuis le plus longtemps
proceed
put(C, CS1, user)
Action
B n'effectue pas d'action dans cet état
return
EOFC(C)
FCC-- (donc FCC=0)
FCC==0 tous les premiers cycles sont terminés

Fig. 152

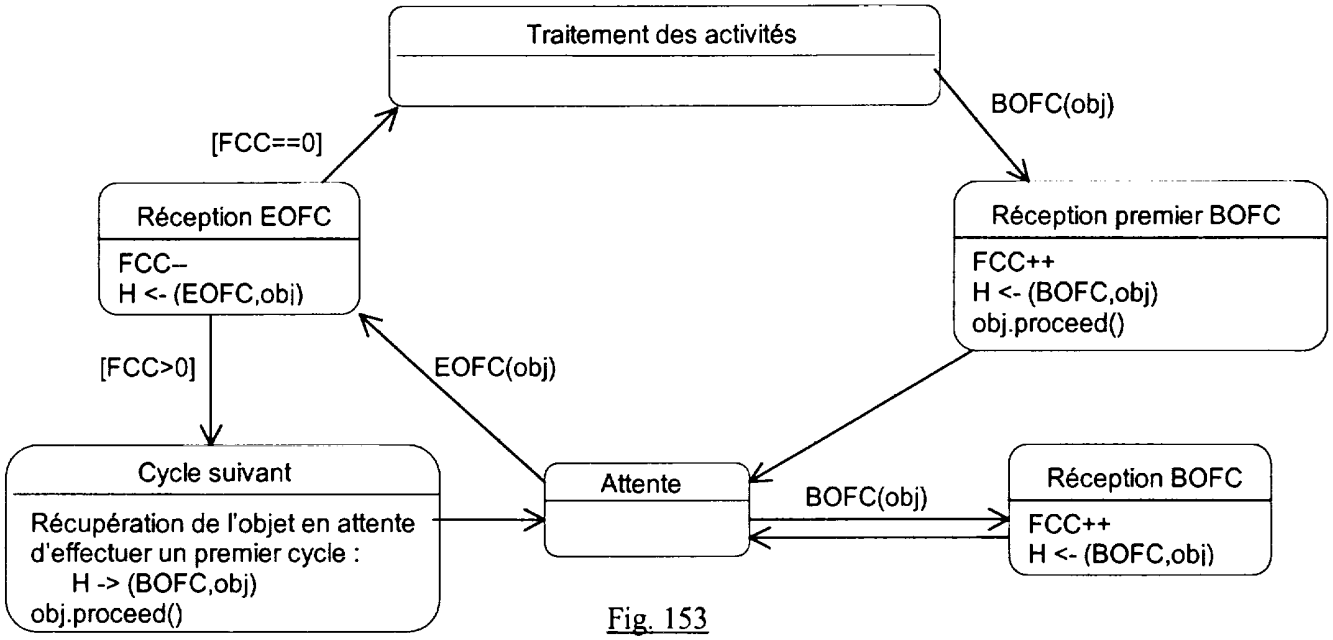


Fig. 153

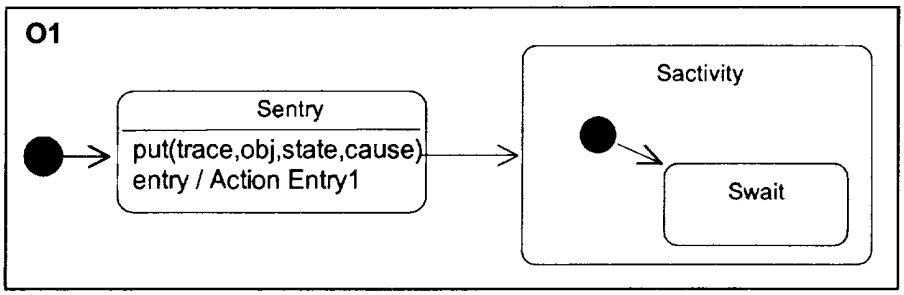


Fig. 154

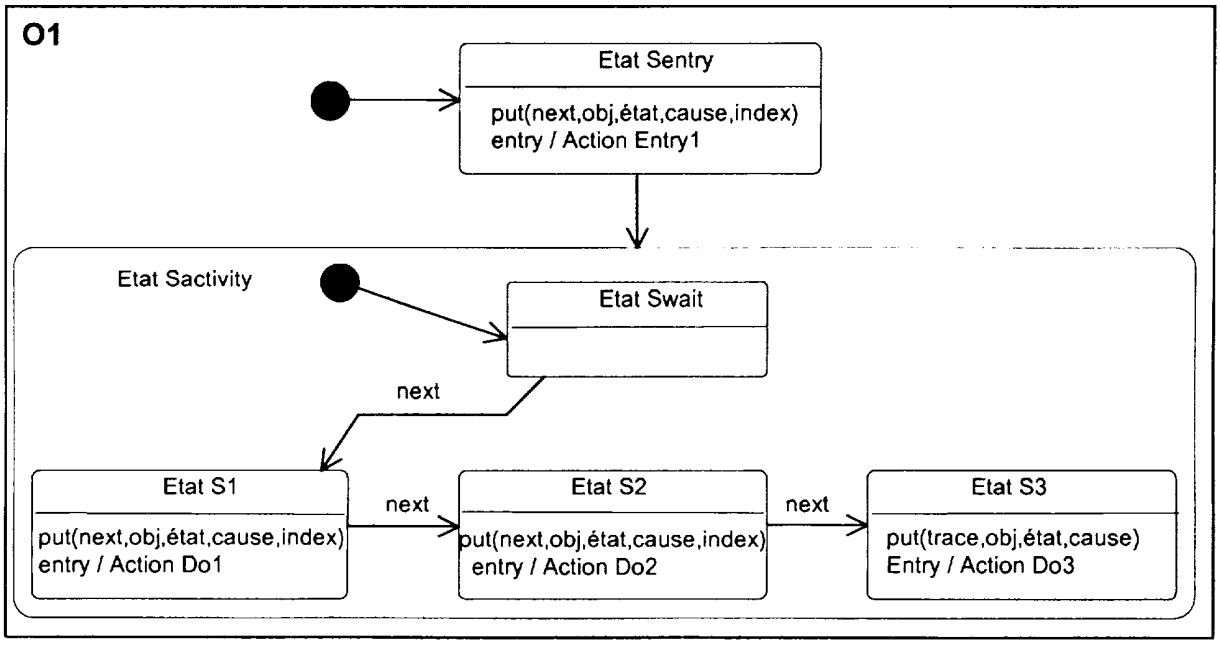


Fig. 155

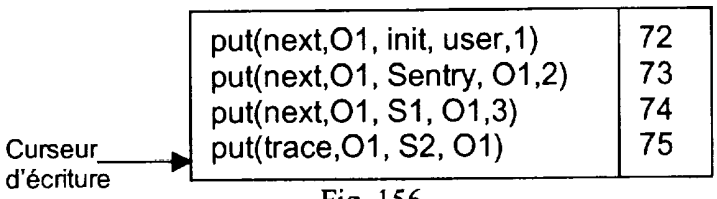


Fig. 156

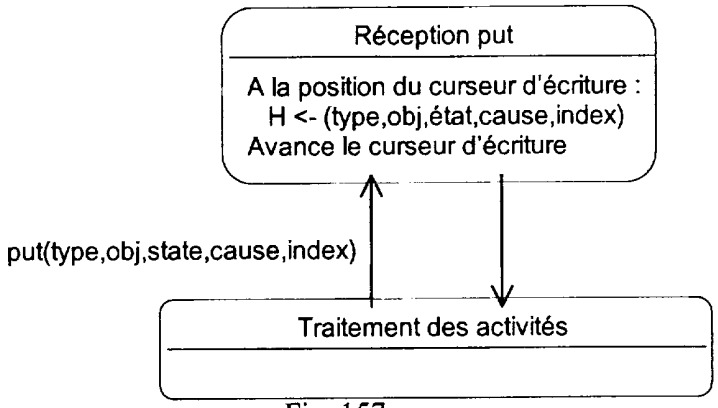


Fig. 157

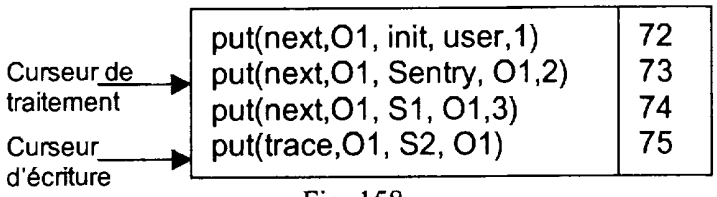


Fig. 158

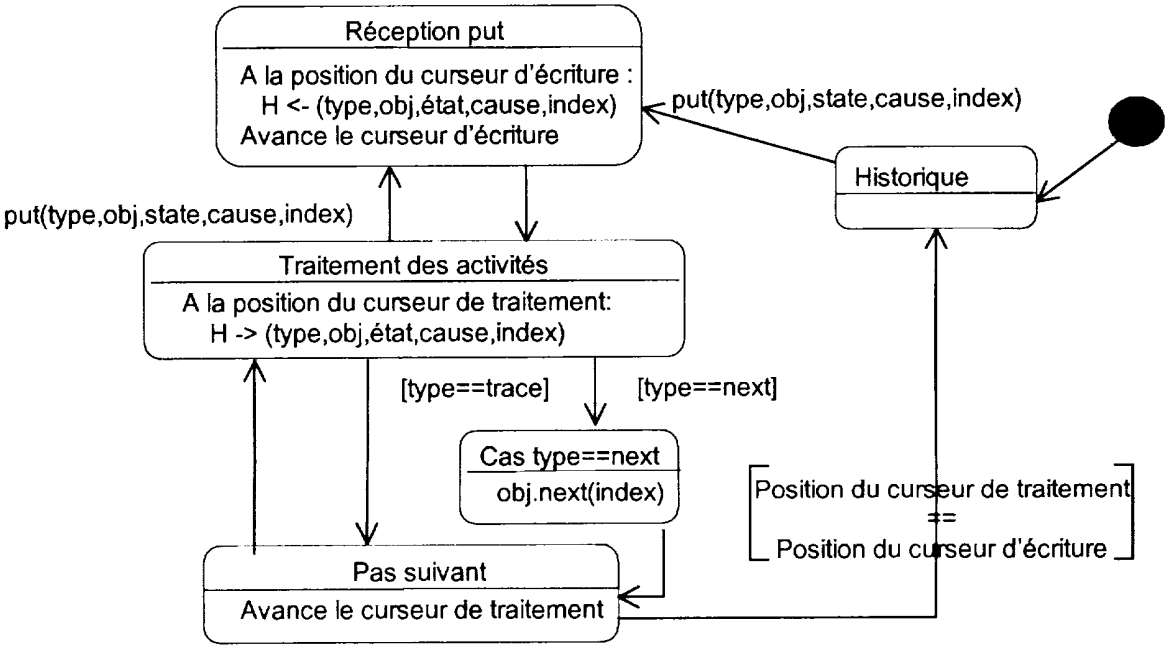


Fig. 159

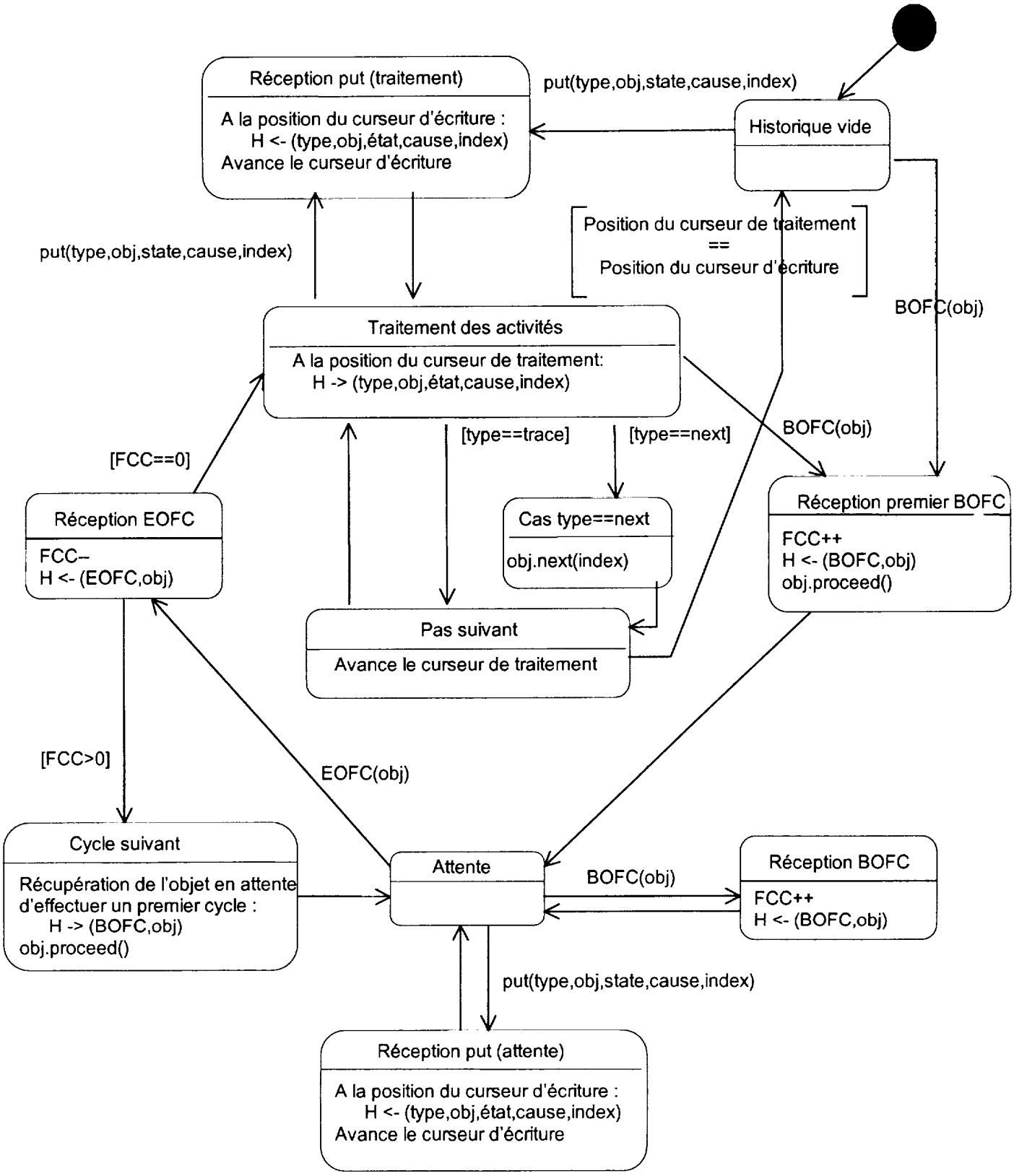

Réception put (traitement)
A la position du curseur d'écriture :
H <- (type,obj,état,cause,index)
Avance le curseur d'écriture
put(type,obj,state,cause,index)
Historique vide
put(type,obj,state,cause,index)
Position du curseur de traitement
==
Position du curseur d'écriture
BOFC(obj)
Traitement des activités
A la position du curseur de traitement:
H -> (type,obj,état,cause,index)
[type==trace]
[type==next]
BOFC(obj)
[FCC==0]
Réception EOFC
FCC--
H <- (EOFC,obj)
Cas type==next
obj.next(index)
Réception premier BOFC
FCC++
H <- (BOFC,obj)
obj.proceed()
Pas suivant
Avance le curseur de traitement
[FCC>0]
EOFC(obj)
Cycle suivant
Récupération de l'objet en attente
d'effectuer un premier cycle :
H -> (BOFC,obj)
obj.proceed()
Attente
BOFC(obj)
Réception BOFC
FCC++
H <- (BOFC,obj)
put(type,obj,state,cause,index)
Réception put (attente)
A la position du curseur d'écriture :
H <- (type,obj,état,cause,index)
Avance le curseur d'écriture

Fig. 160

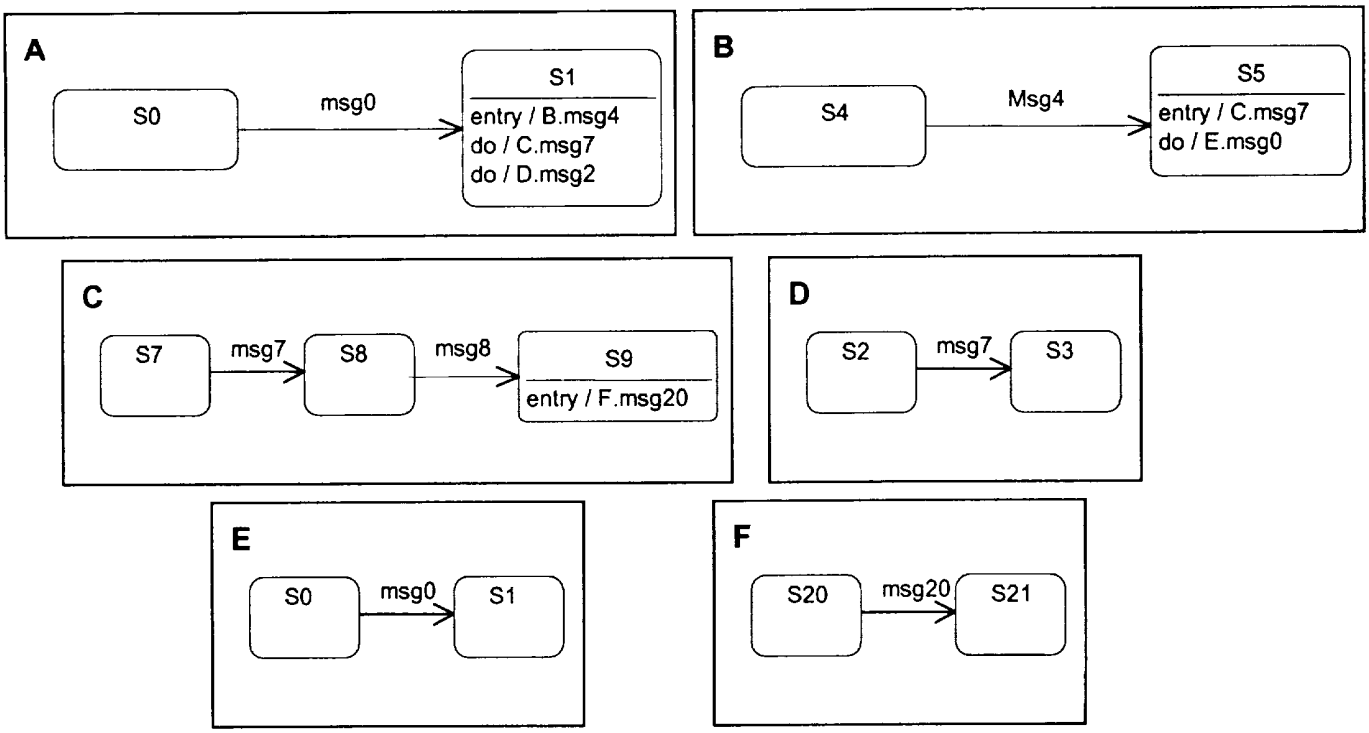

A
S0
msg0
S1
entry / B.msg4
do / C.msg7
do / D.msg2
B
S4
Msg4
S5
entry / C.msg7
do / E.msg0
C
S7
msg7
S8
msg8
S9
entry / F.msg20
D
S2
msg7
S3
E
S0
msg0
S1
F
S20
msg20
S21

Fig. 161

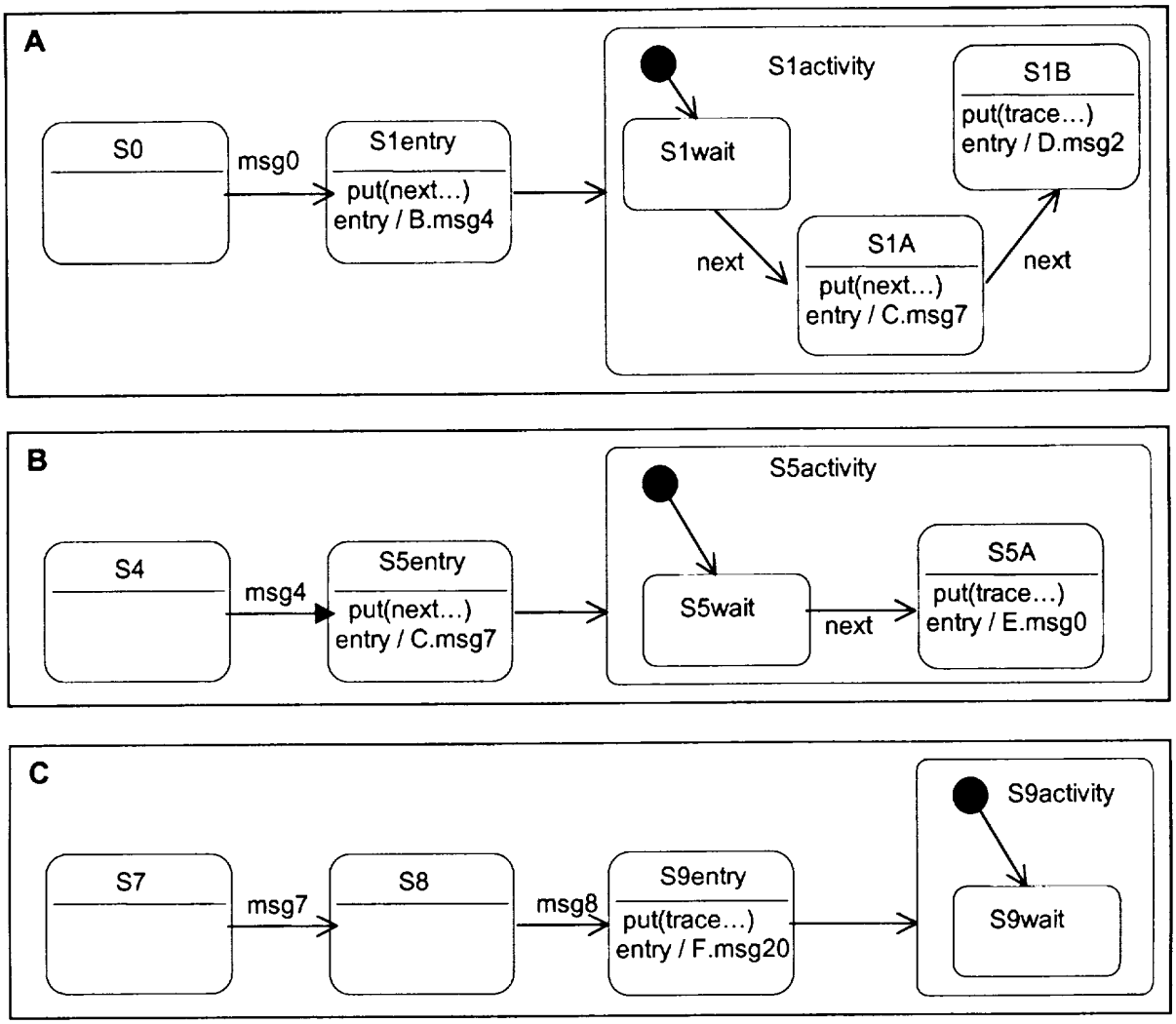

A
S0
msg0
S1entry
put(next...)
entry / B.msg4
S1activity
S1wait
S1B
put(trace...)
entry / D.msg2
next
S1A
put(next...)
entry / C.msg7
next
B
S4
msg4
S5entry
put(next...)
entry / C.msg7
S5activity
S5wait
next
S5A
put(trace...)
entry / E.msg0
C
S7
msg7
S8
msg8
S9entry
put(trace...)
entry / F.msg20
S9activity
S9wait

Fig. 162

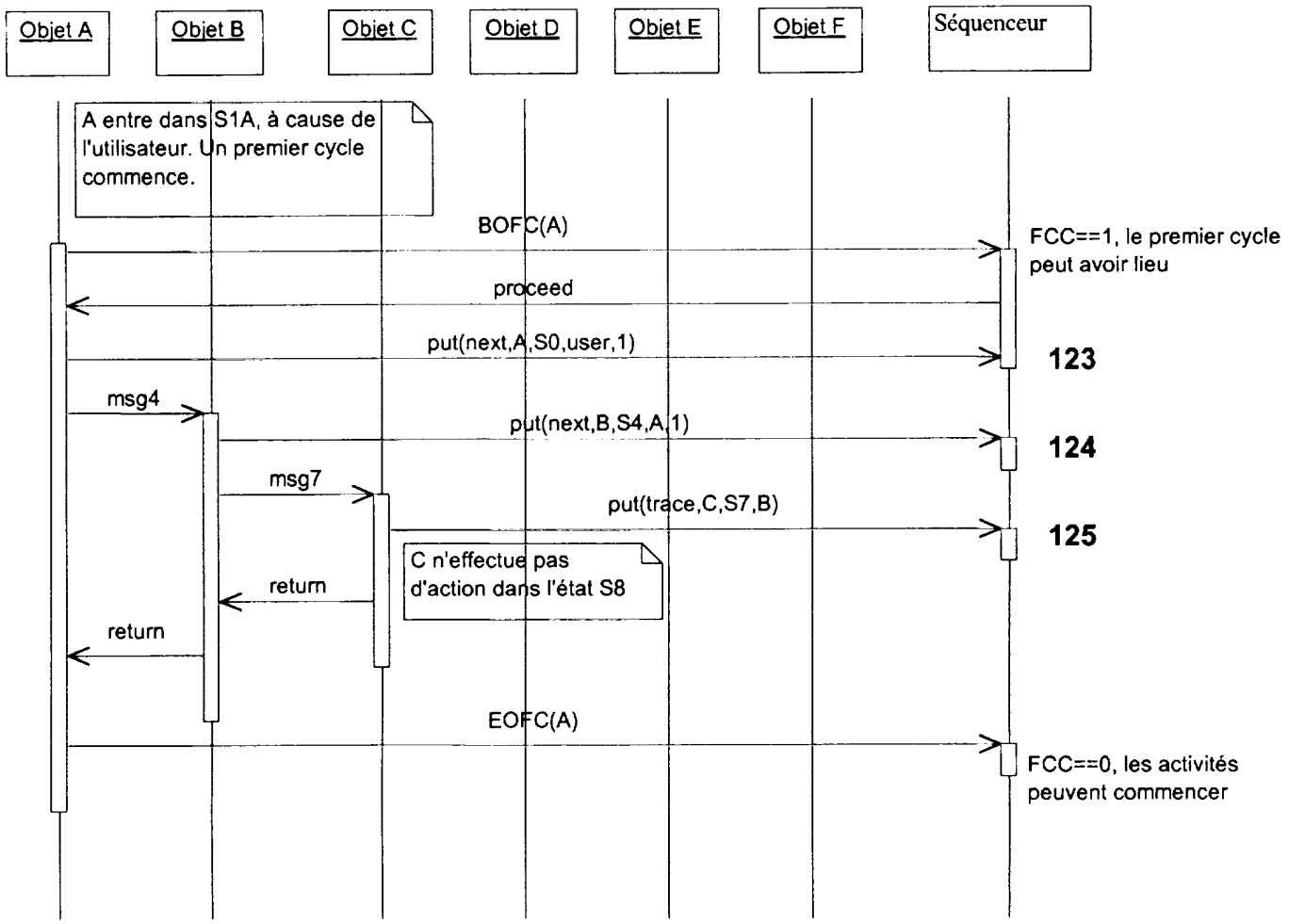

Objet A
Objet B
Objet C
Objet D
Objet E
Objet F
Séquenceur
A entre dans S1A, à cause de l'utilisateur. Un premier cycle commence.
BOFC(A)
FCC==1, le premier cycle peut avoir lieu
proceed
put(next,A,S0,user,1)
123
msg4
put(next,B,S4,A,1)
124
msg7
put(trace,C,S7,B)
125
C n'effectue pas d'action dans l'état S8
return
return
EOFC(A)
FCC==0, les activités peuvent commencer

Fig. 163

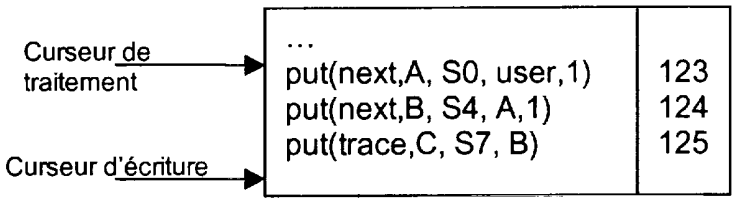


Fig. 164

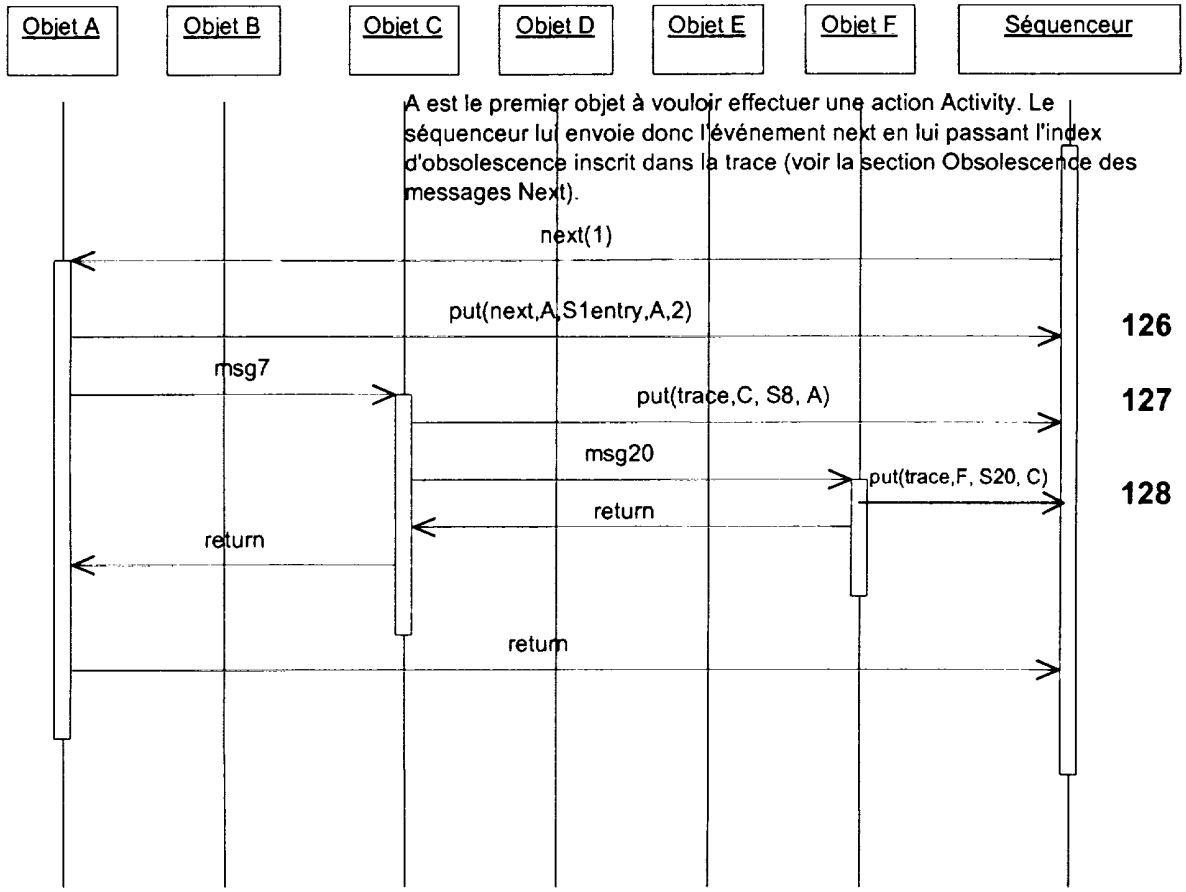


Fig. 165

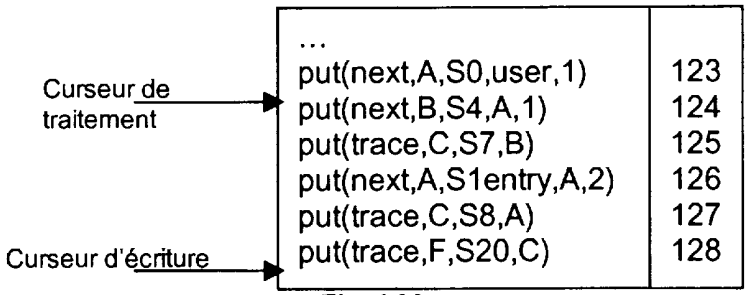


Fig. 166

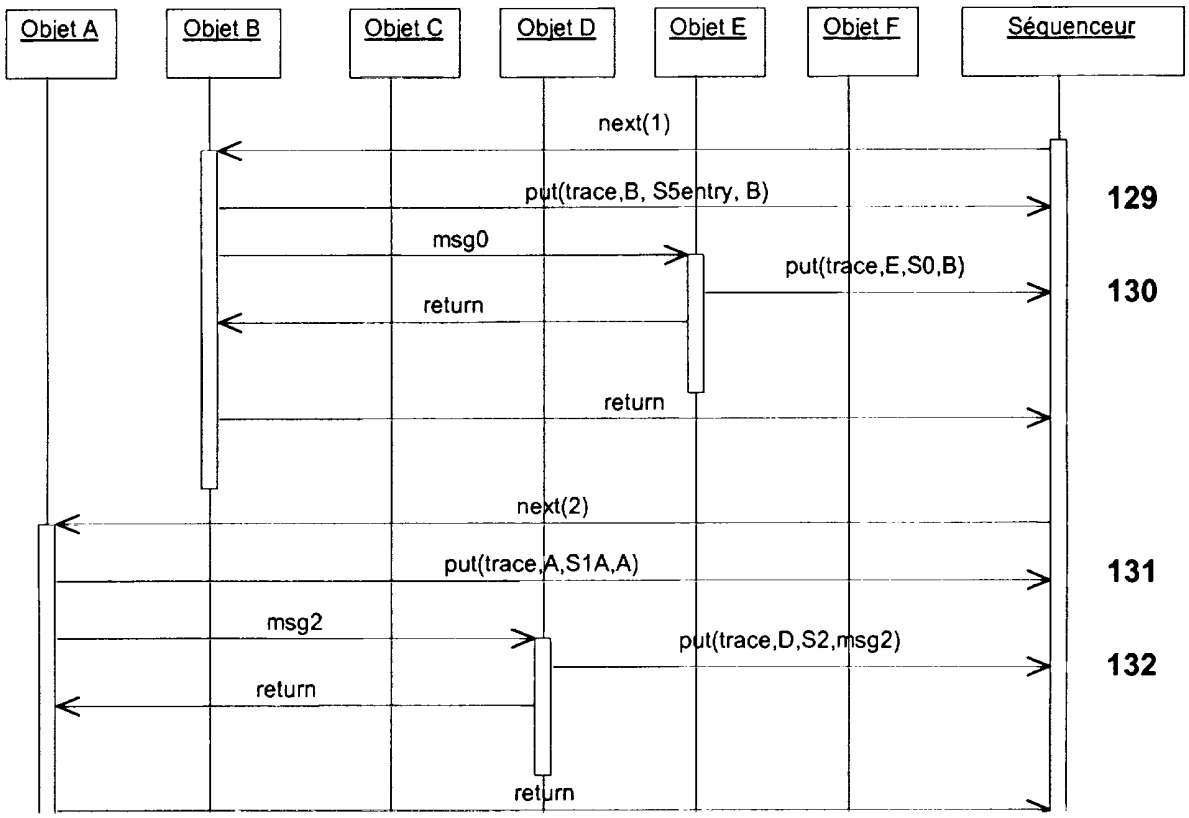


Fig. 167

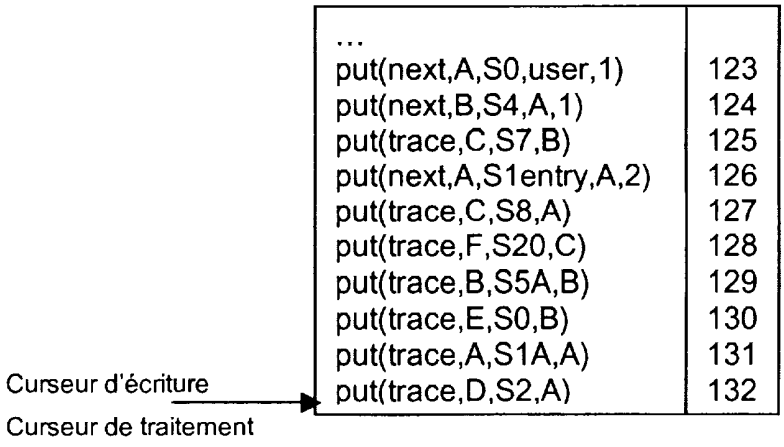


Fig. 168

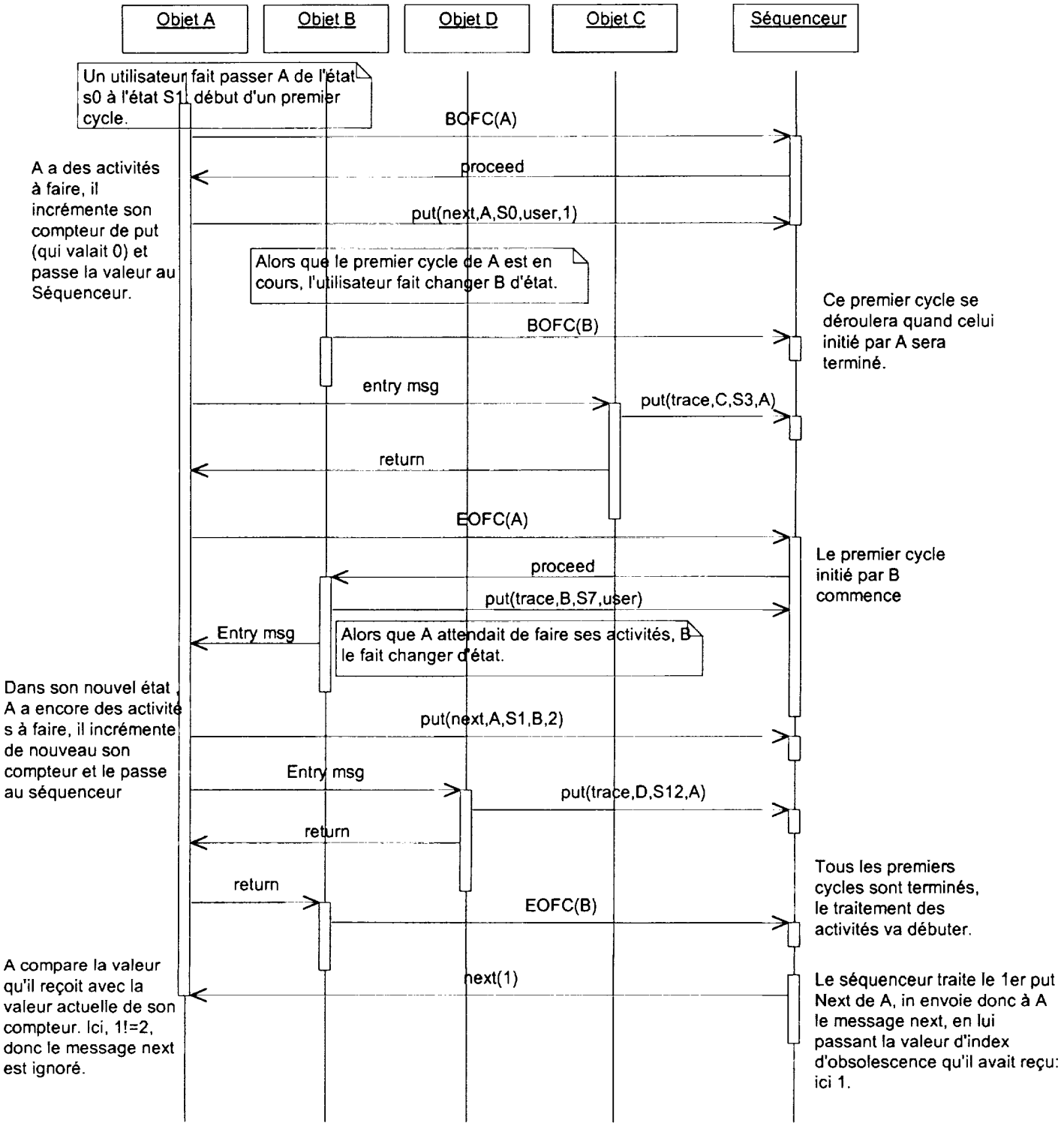

Objet A
Objet B
Objet D
Objet C
Séquenceur
Un utilisateur fait passer A de l'état s0 à l'état S1, début d'un premier cycle.
BOFC(A)
proceed
A a des activités à faire, il incrémente son compteur de put (qui valait 0) et passe la valeur au Séquenceur.
put(next,A,S0,user,1)
Alors que le premier cycle de A est en cours, l'utilisateur fait changer B d'état.
Ce premier cycle se déroulera quand celui initié par A sera terminé.
BOFC(B)
entry msg
put(trace,C,S3,A)
return
EOFC(A)
Le premier cycle initié par B commence
proceed
put(trace,B,S7,user)
Entry msg
Alors que A attendait de faire ses activités, B le fait changer d'état.
Dans son nouvel état, A a encore des activités à faire, il incrémente de nouveau son compteur et le passe au séquenceur
put(next,A,S1,B,2)
Entry msg
put(trace,D,S12,A)
return
Tous les premiers cycles sont terminés, le traitement des activités va débuter.
return
EOFC(B)
A compare la valeur qu'il reçoit avec la valeur actuelle de son compteur. Ici, 1!=2, donc le message next est ignoré.
next(1)
Le séquenceur traite le 1er put Next de A, in envoie donc à A le message next, en lui passant la valeur d'index d'obsolescence qu'il avait reçu: ici 1.

Fig. 169

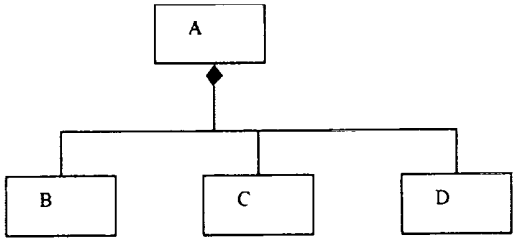


Fig. 170

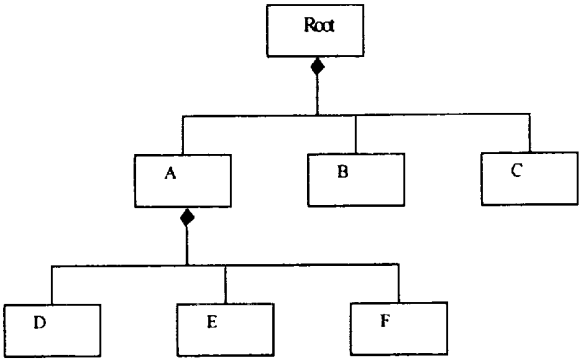


Fig. 171

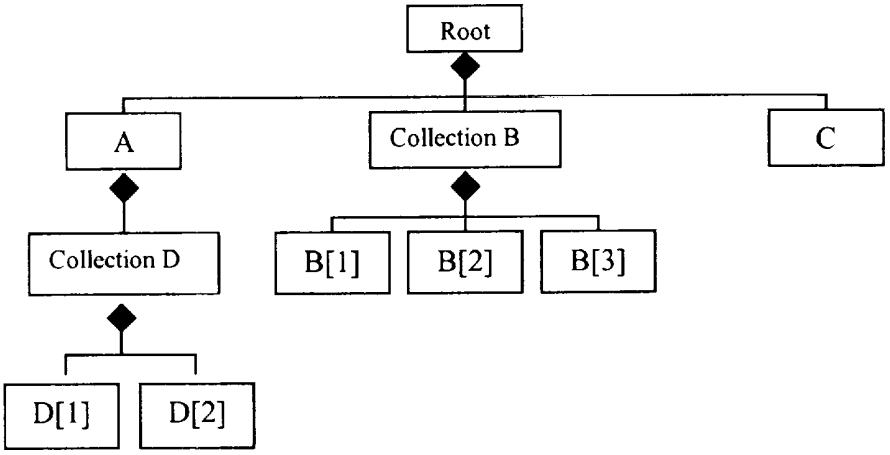


Fig. 172

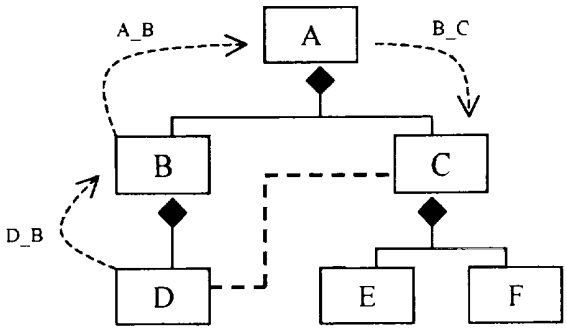


Fig. 173

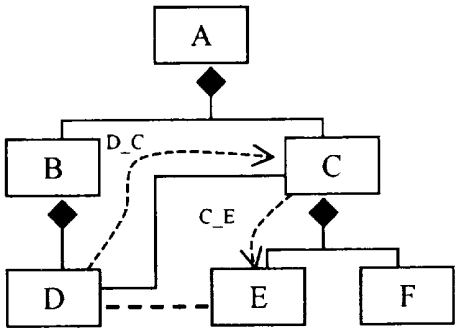


Fig. 174

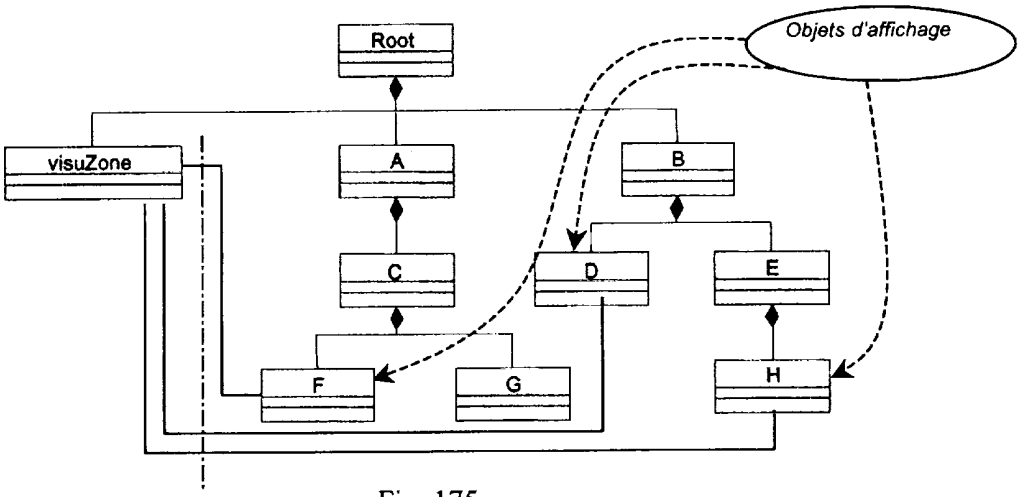


Fig. 175

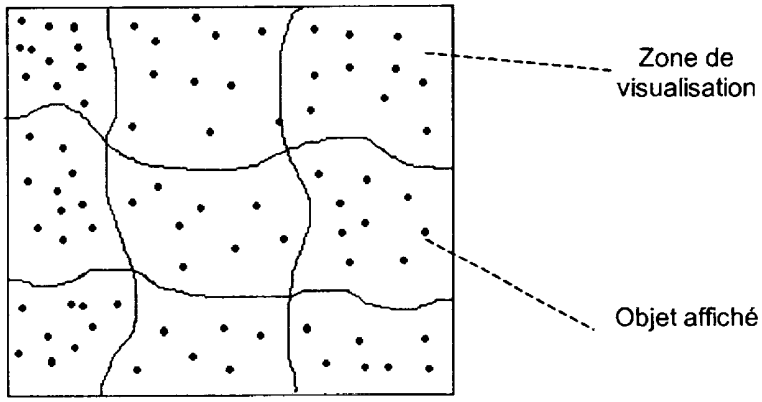


Fig. 176

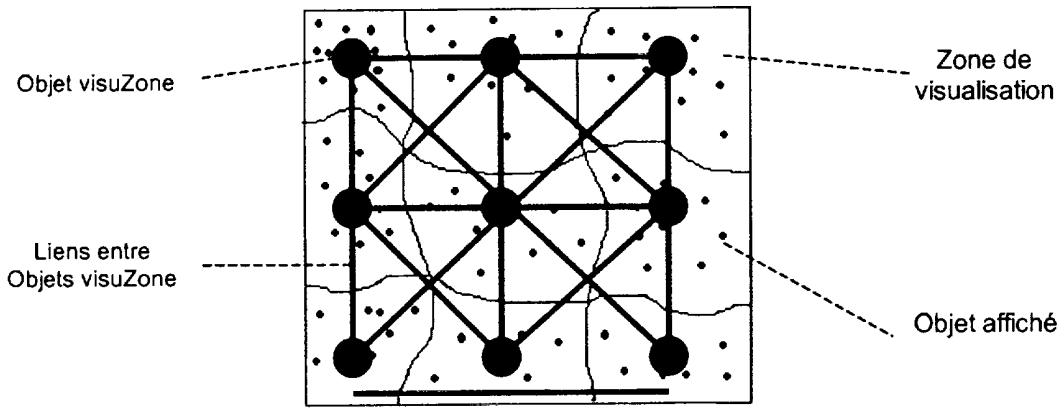


Fig. 177

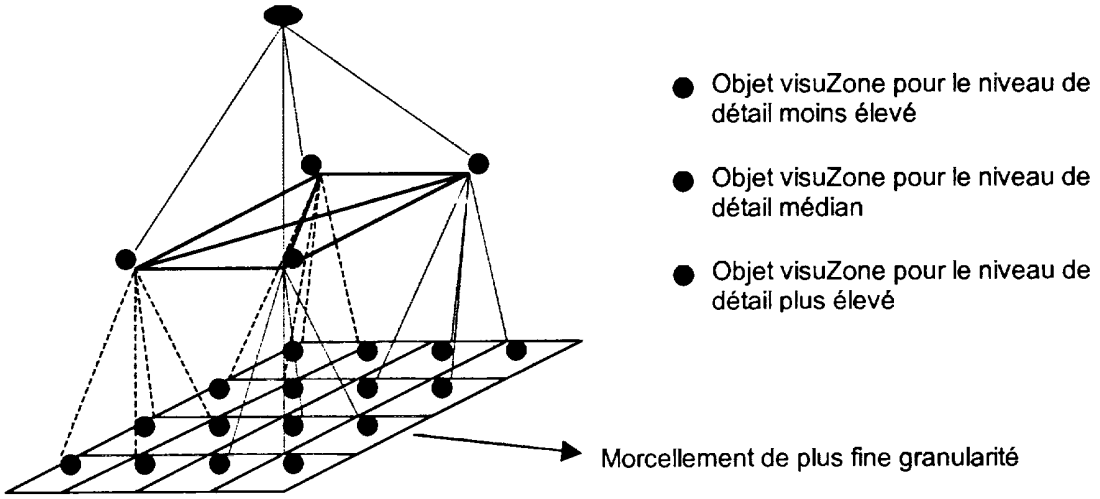


Fig. 178

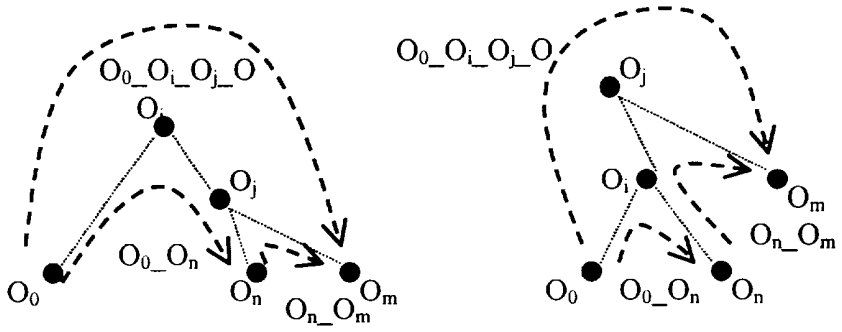


Fig. 179

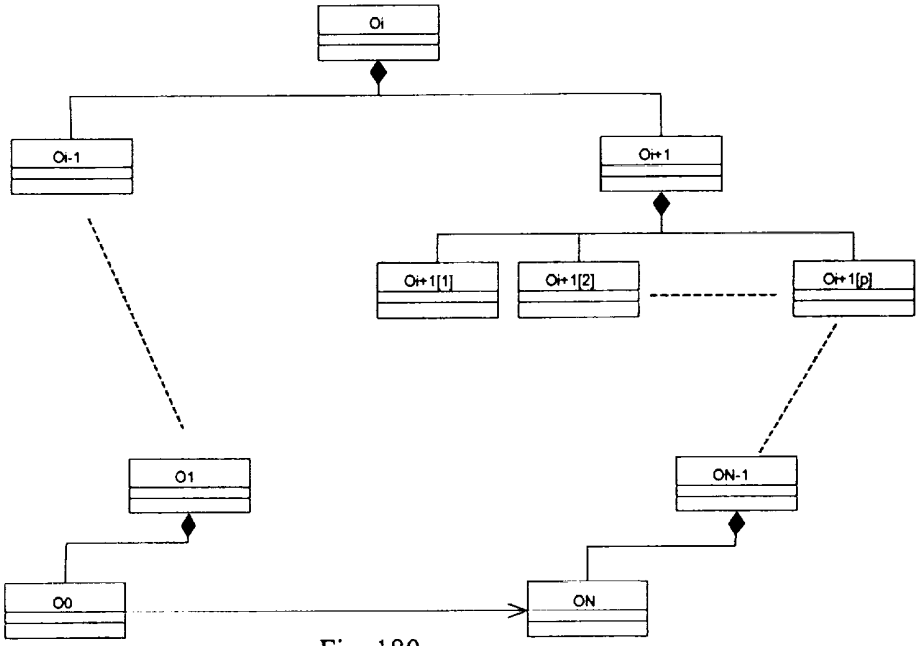


Fig. 180

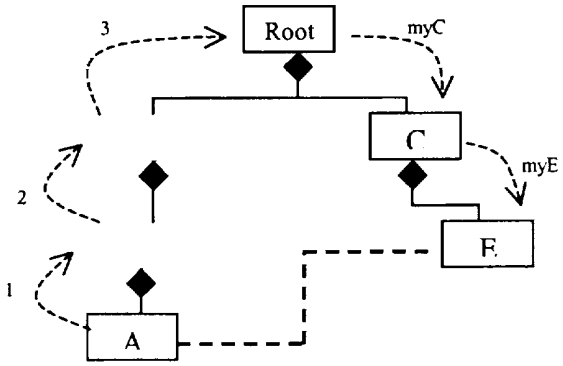


Fig. 181

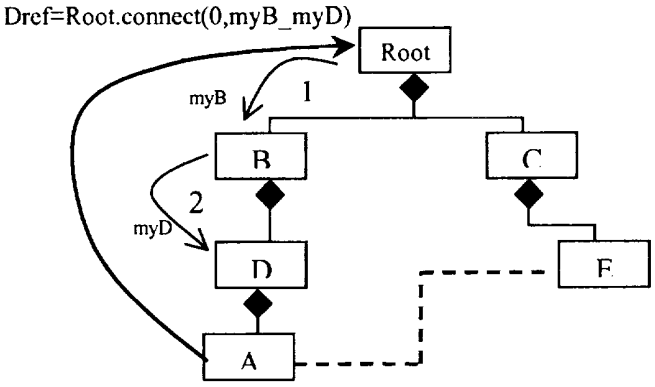


Fig. 182

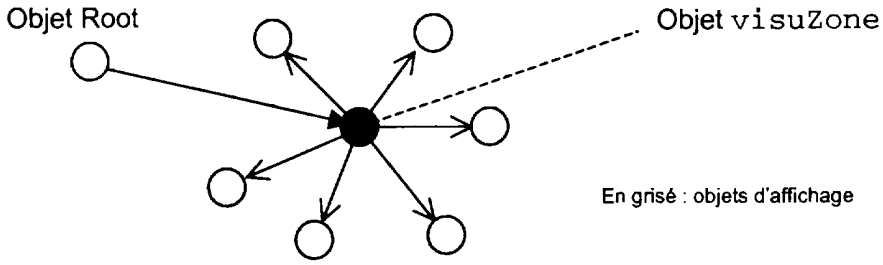


Fig. 183

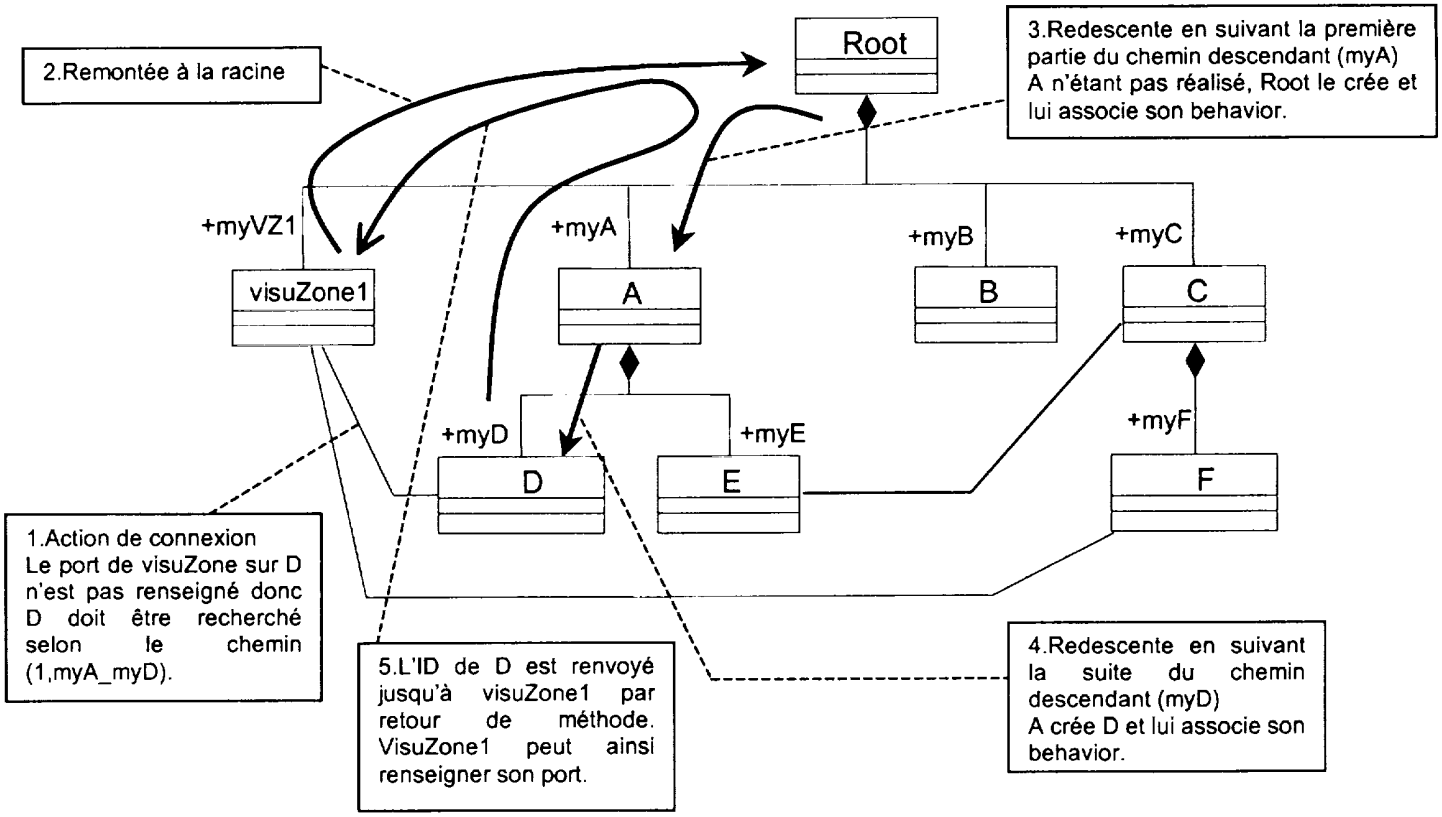


Fig. 184

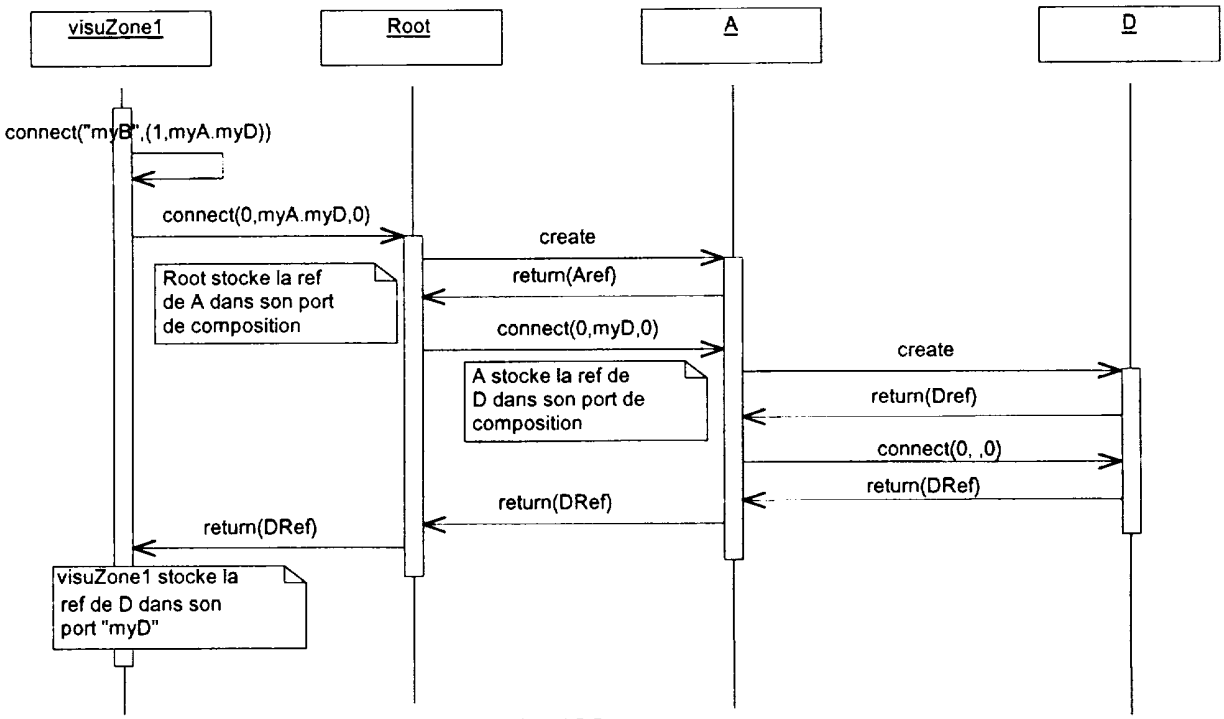


Fig. 185

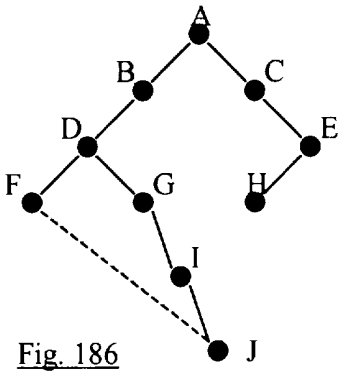


Fig. 186

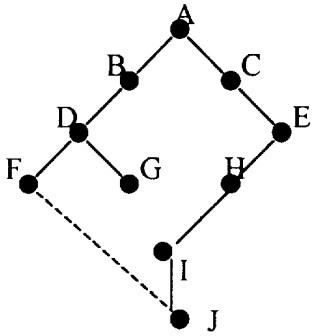


Fig. 187

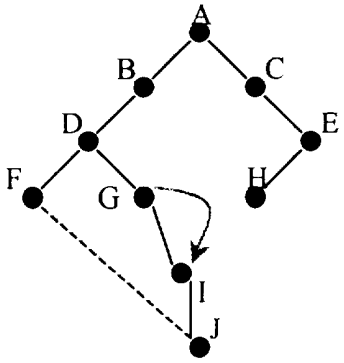


Fig. 188

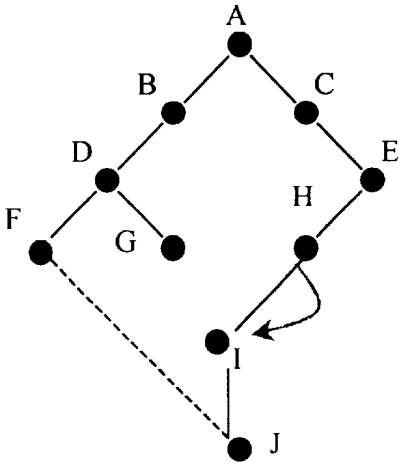


Fig. 189

put(trace,A,S0,user)	t
put(trace,B,S4,A)	t+1
put(trace,C,S12,user)	t+2
put(trace,D,S2,B)	t+3
put(trace,B,S5,C)	t+4
put(trace,E,S0,D)	t+5
put(trace,C,S13,B)	t+6
put(trace,F,S20,E)	t+7
put(trace,B,S6,C)	t+8
...	

Fig. 190

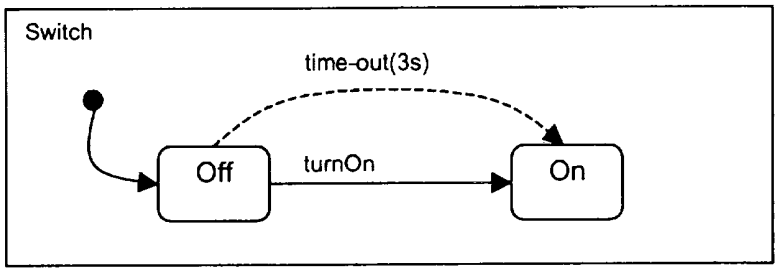


Fig. 191

Connexion à la profondeur 3 :

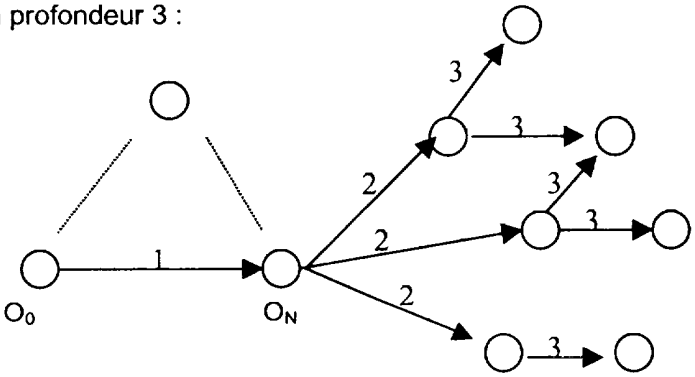


Fig. 192

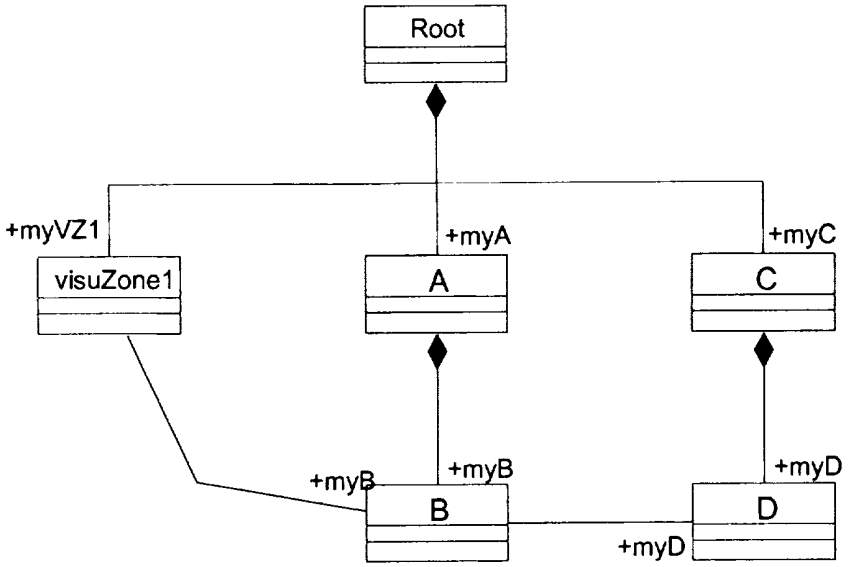


Fig. 193

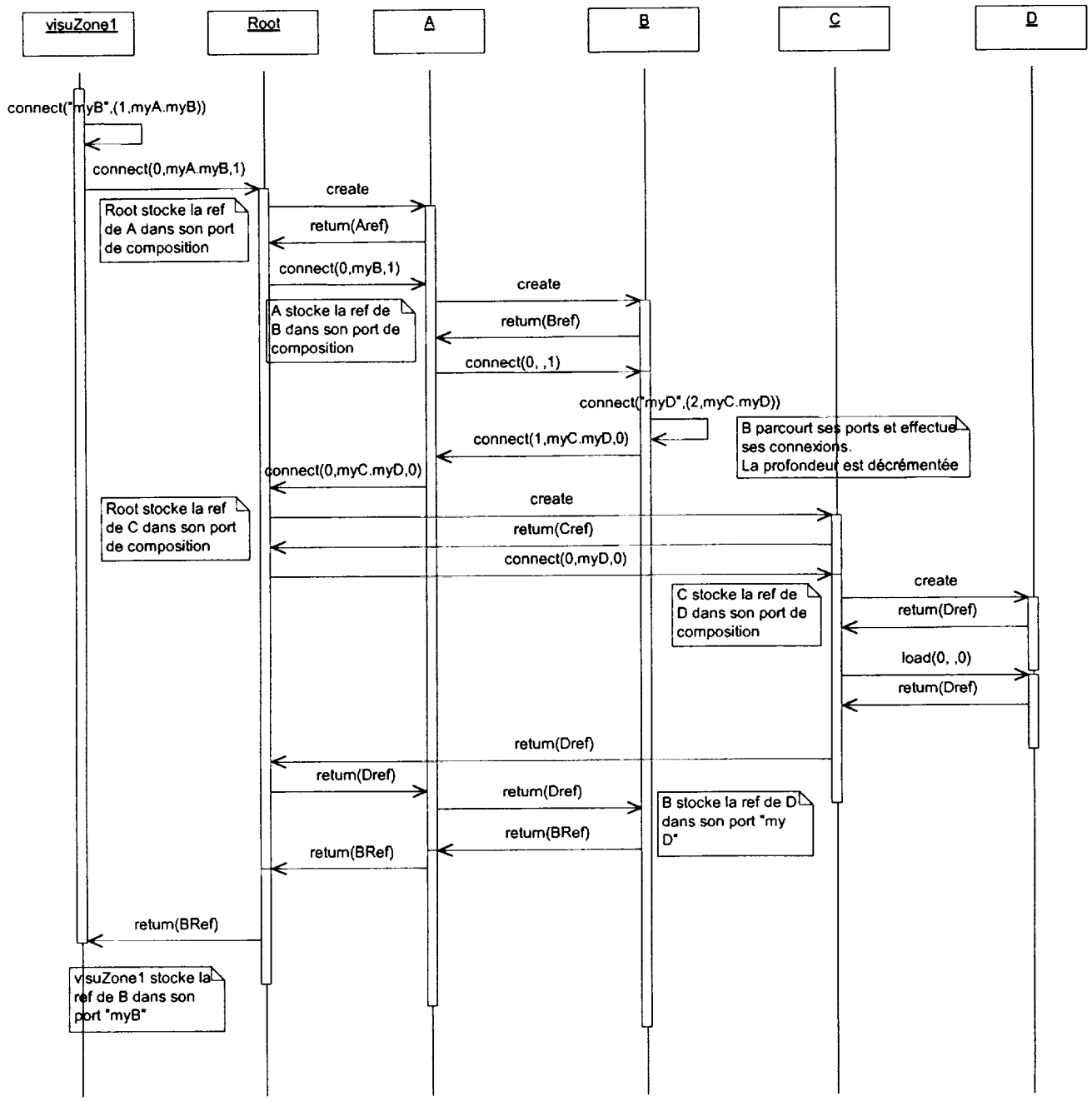

visuZone1
Root
A
B
C
D
connect("myB",(1,myA.myB))
connect(0,myA.myB,1)
create
Root stocke la ref de A dans son port de composition
return(Aref)
connect(0,myB,1)
create
A stocke la ref de B dans son port de composition
return(Bref)
connect(0, ,1)
connect("myD",(2,myC.myD))
B parcourt ses ports et effectue ses connexions. La profondeur est décrémentée
connect(1,myC.myD,0)
connect(0,myC.myD,0)
create
Root stocke la ref de C dans son port de composition
return(Cref)
connect(0,myD,0)
create
C stocke la ref de D dans son port de composition
return(Dref)
load(0, ,0)
return(Dref)
return(Dref)
return(Dref)
return(Dref)
B stocke la ref de D dans son port "my D"
return(BRef)
return(BRef)
return(BRef)
visuZone1 stocke la ref de B dans son port "myB"

Fig. 194

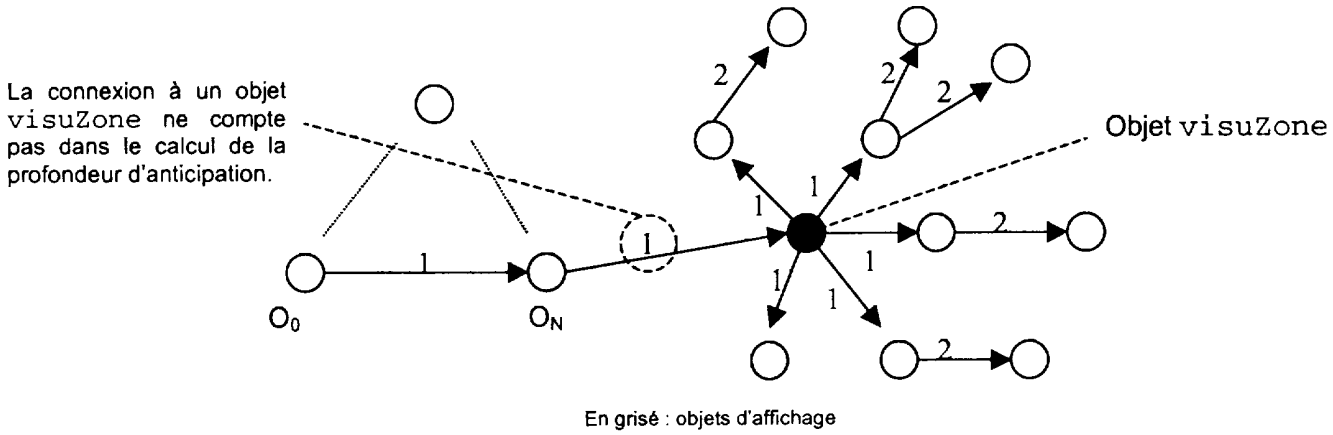


Fig. 195

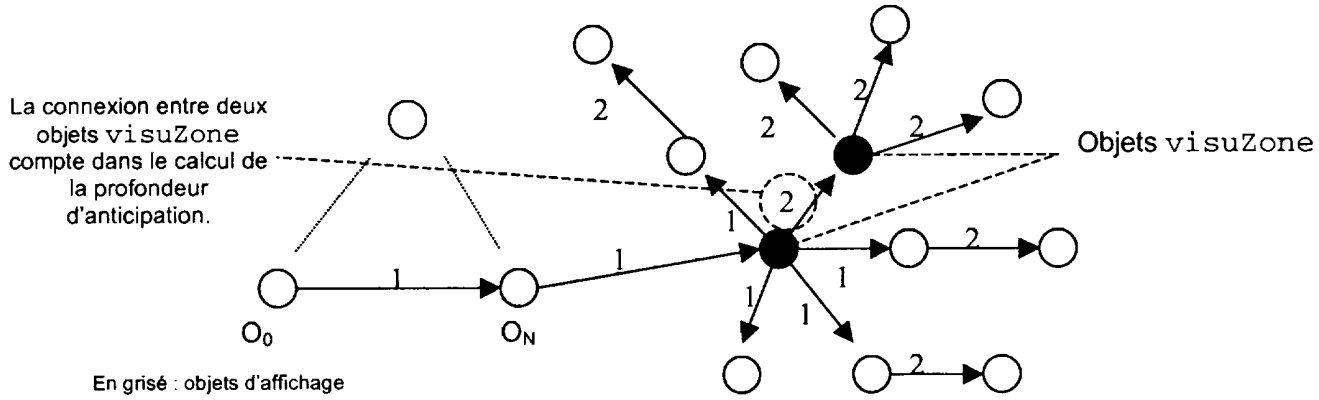


Fig. 196

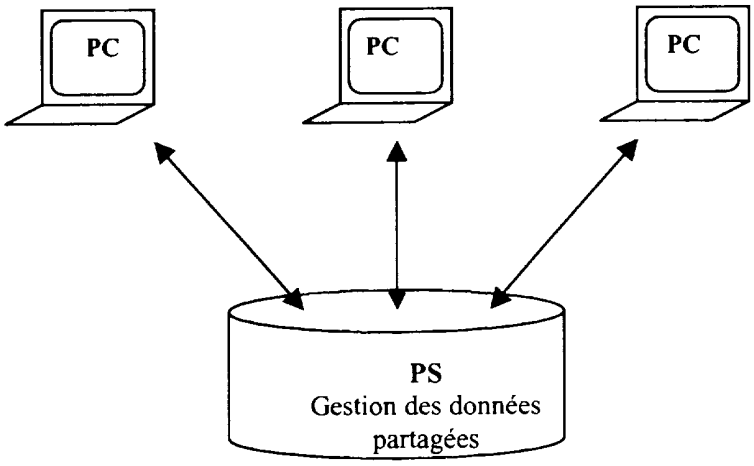


Fig. 197

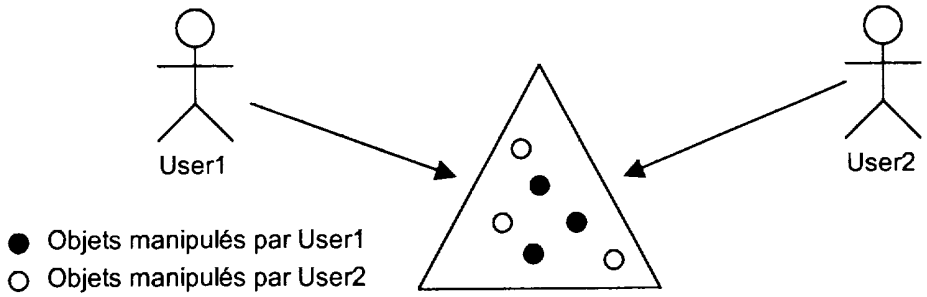


Fig. 198

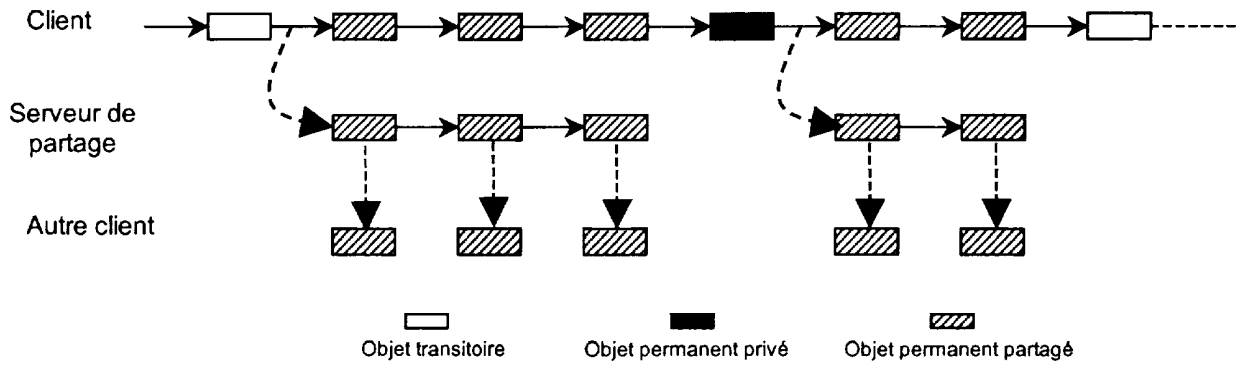


Fig. 199

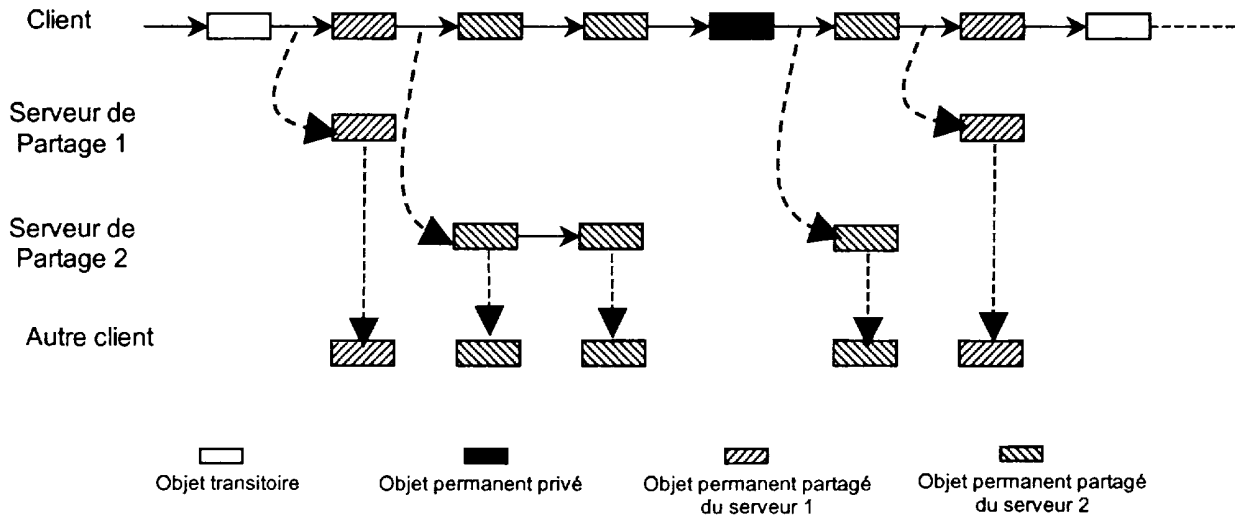


Fig. 200

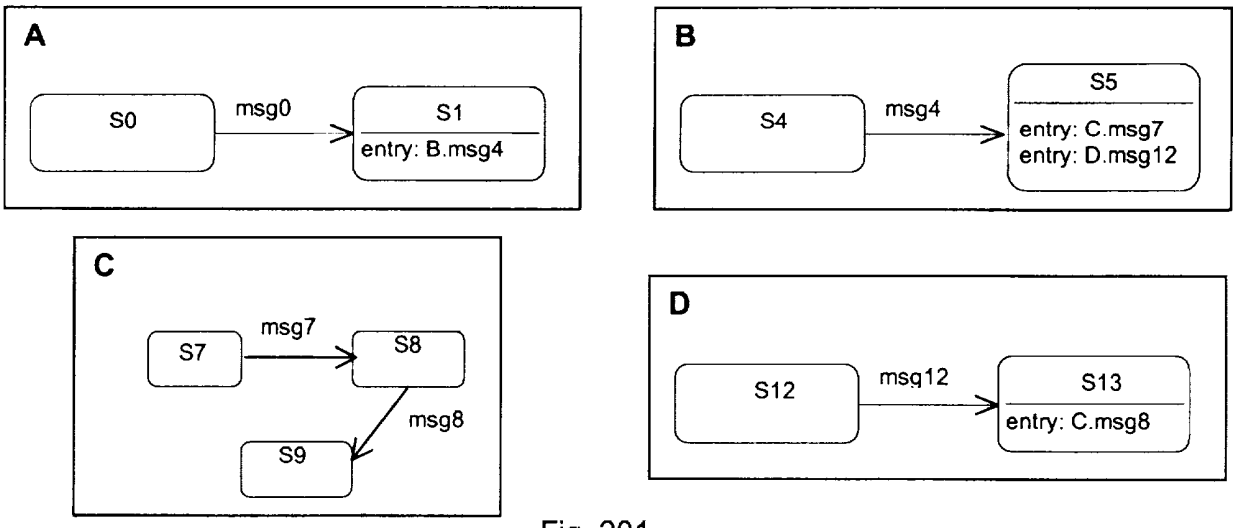


Fig. 201

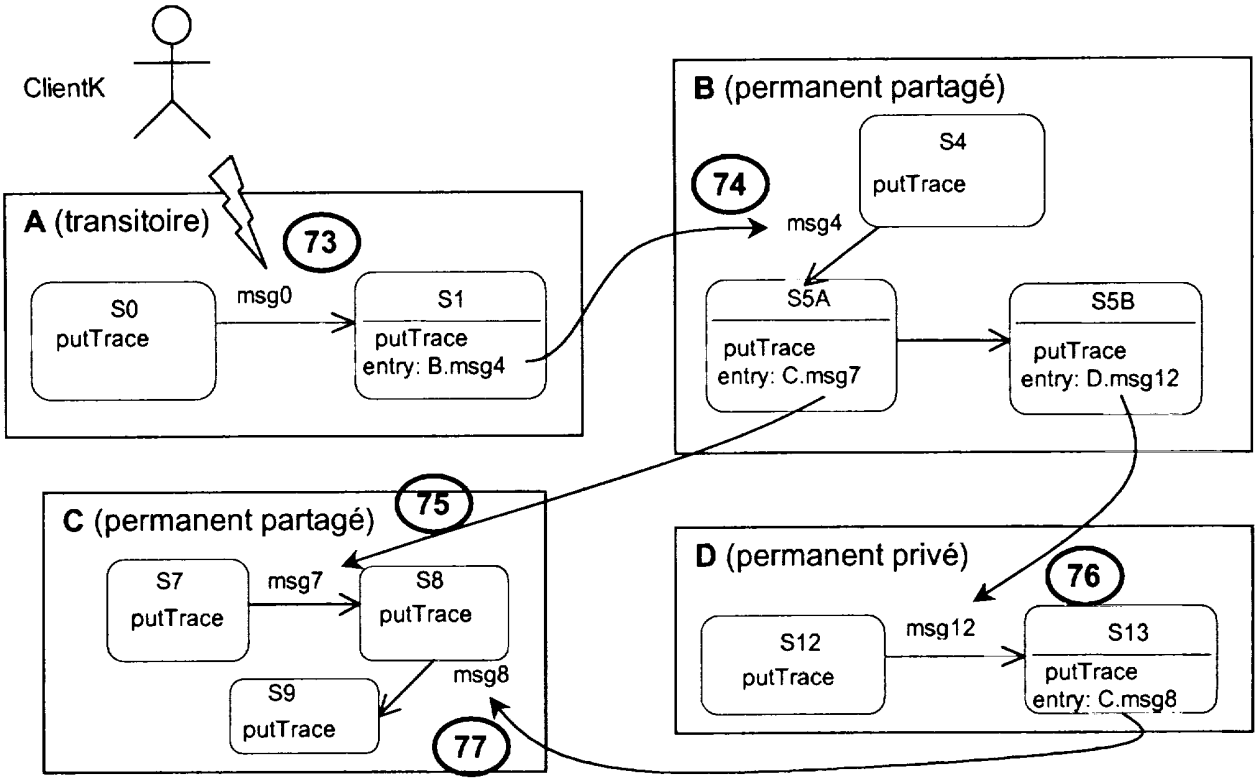


Fig. 202

B (partagé) effectue une transition due à un objet transitoire. Cette transition est répliquée sur le serveur.

ClientK

put(trace,A,S0,user)	72
put(trace,B,S4,A)	73
put(trace,C,S7,B)	74
put(trace,B,S5A,B)	75
put(trace,D,S12,B)	76
put(trace,C,S8,D)	77

Serveur

put(trace,B,S4,client)	ClientK,73
put(trace,C,S7,B)	ClientK,74
put(trace,B,S5A,B)	ClientK,75
put(trace,C,S8,client)	ClientK,77

Comme une application tourne sur le serveur, les transitions des objets partagés se font directement sur le serveur, indépendamment de

D est un objet permanent privé, l'application du serveur n'en a pas la visibilité. Le message envoyé par B dans l'état S5B est ignoré !

C (partagé) effectue une transition due à un objet permanent privé. Cette transition est répliquée sur le serveur.

Fig. 203

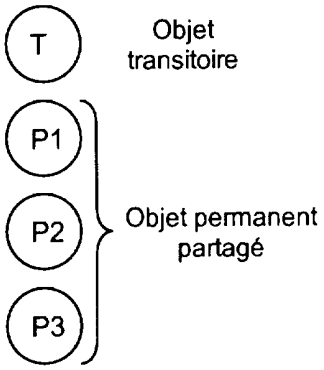
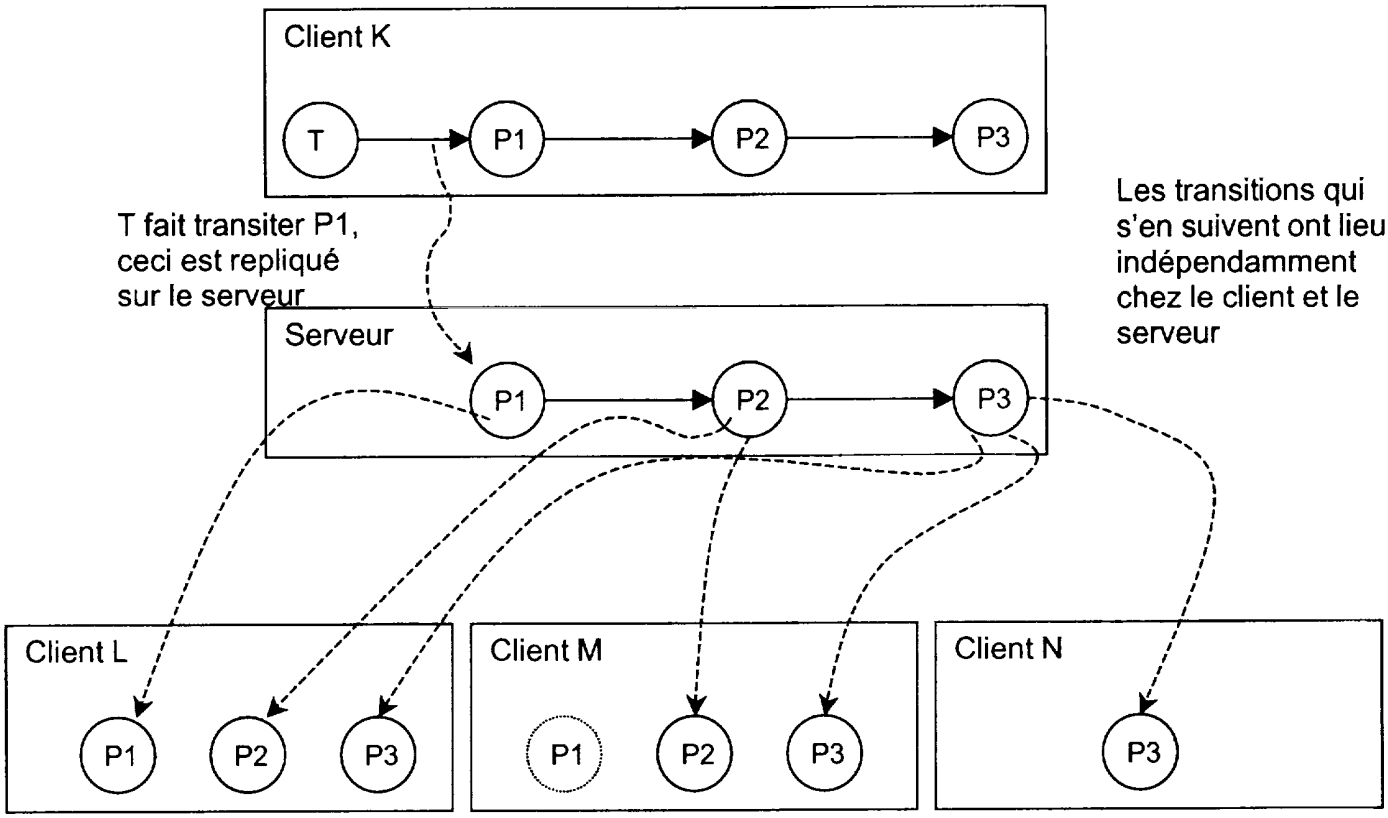


Fig. 204



Le client L possède réellement P1 P2 et P3. Les transitions de chacun de ces objets sont répliquées depuis le serveur

Le client M ne possède réellement que P2 et P3. Leurs transitions sont répliquées. P1 n'est là que virtuellement, sa transition n'est pas répliquée. depuis le serveur

Seul P3 est présent chez le client N, sa transition est répliquée.

Fig. 205

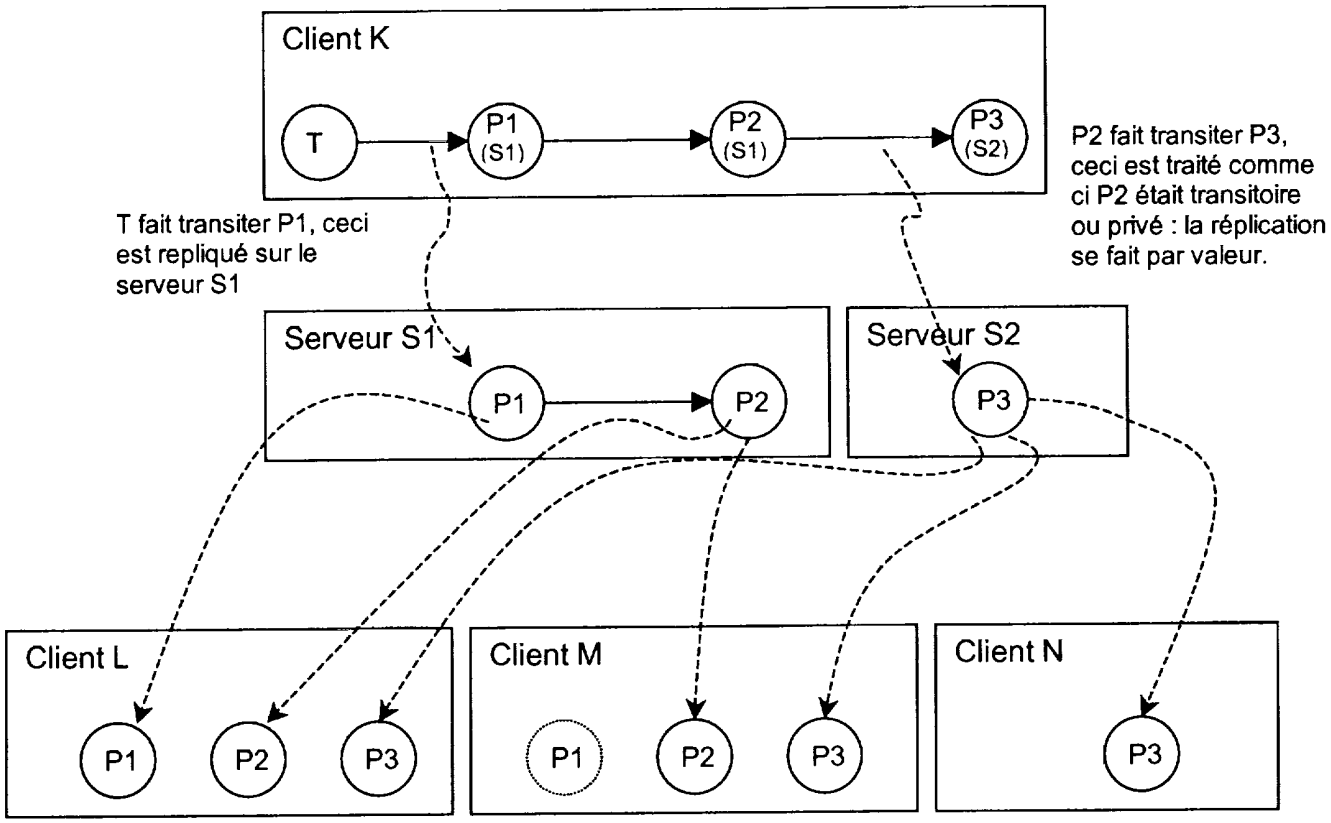


Fig. 206

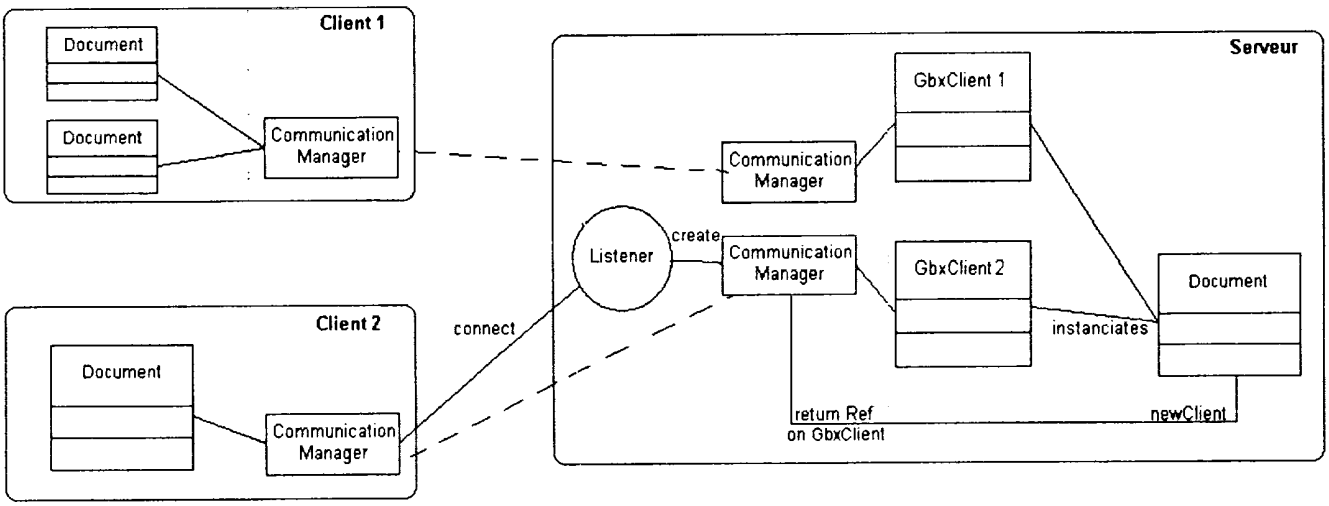


Fig. 207

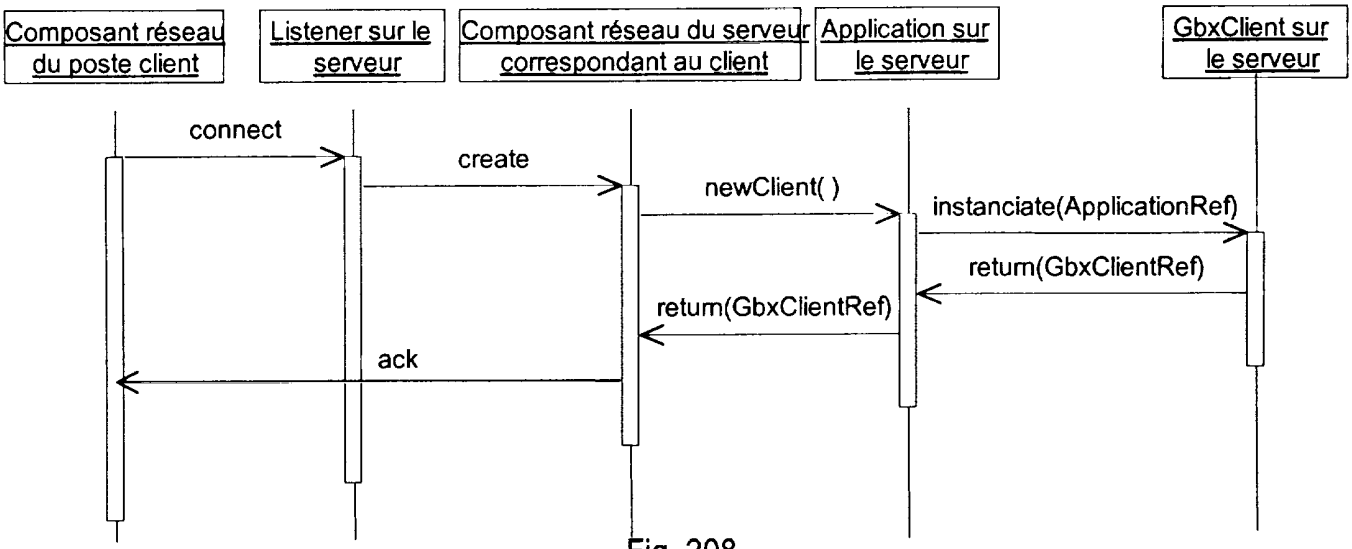


Fig. 208

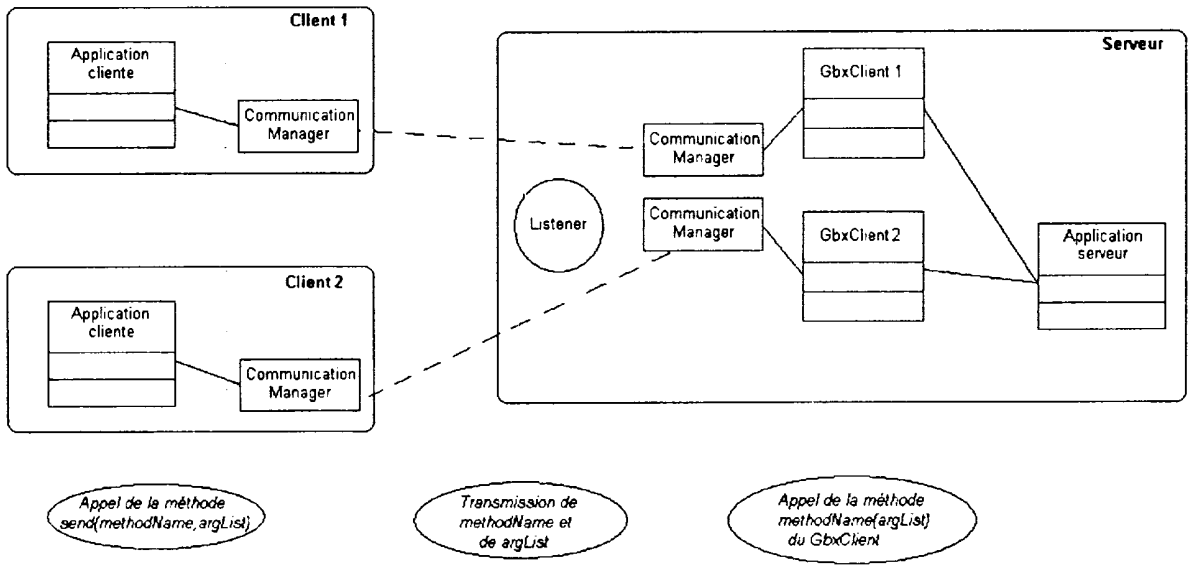


Fig. 209

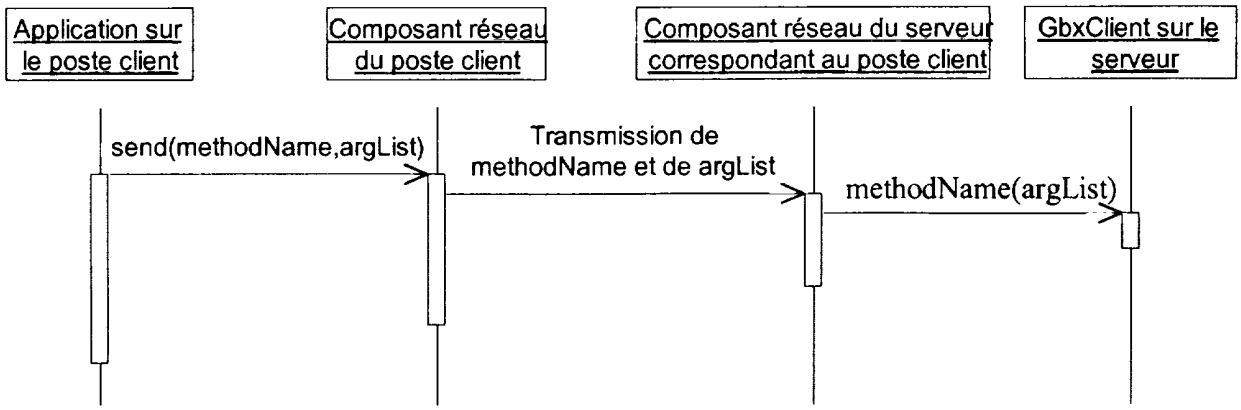


Fig. 210

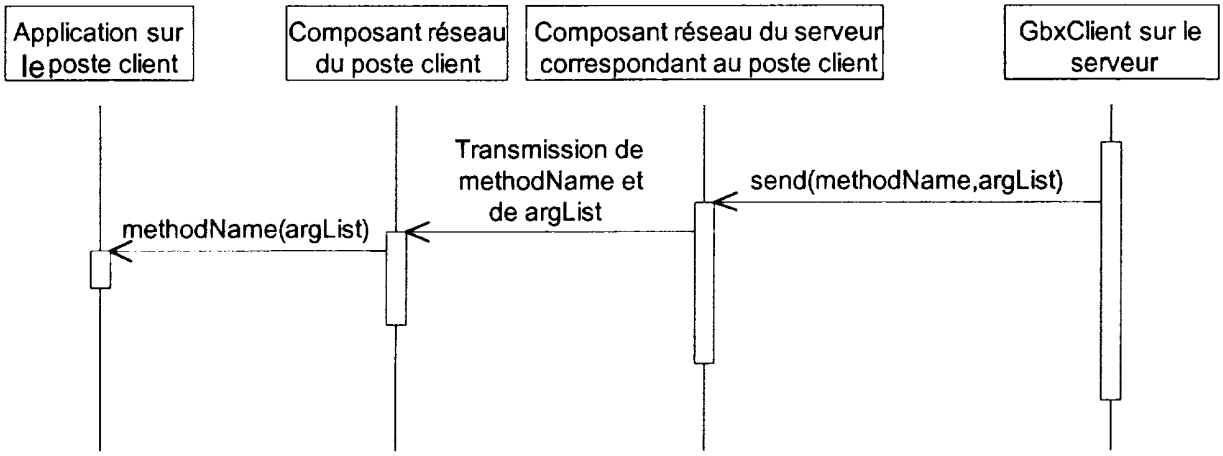


Fig. 211

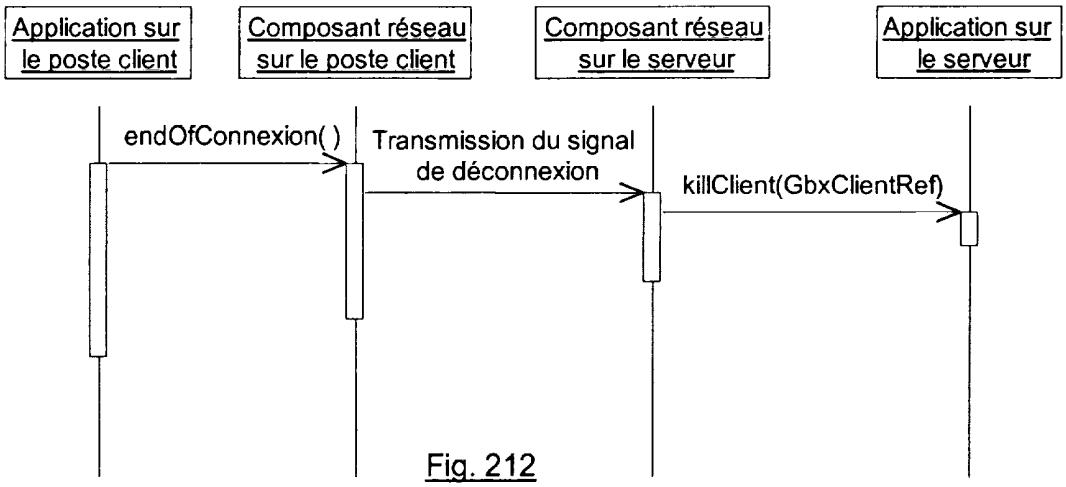


Fig. 212

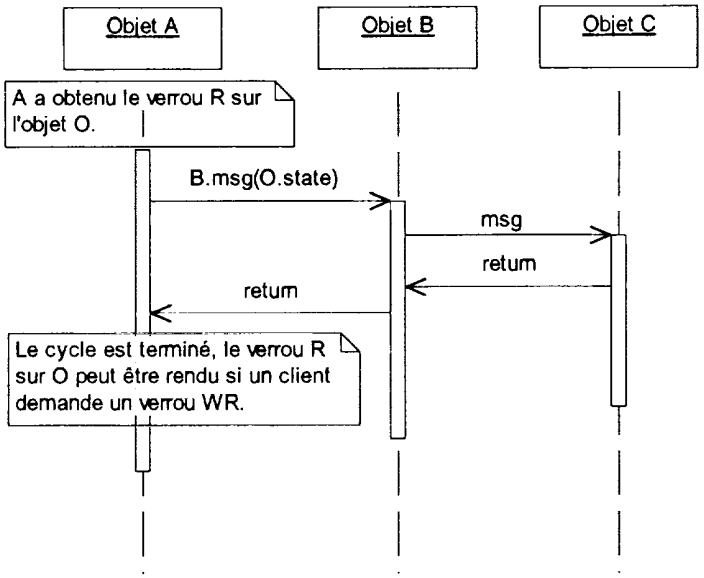


Fig. 213

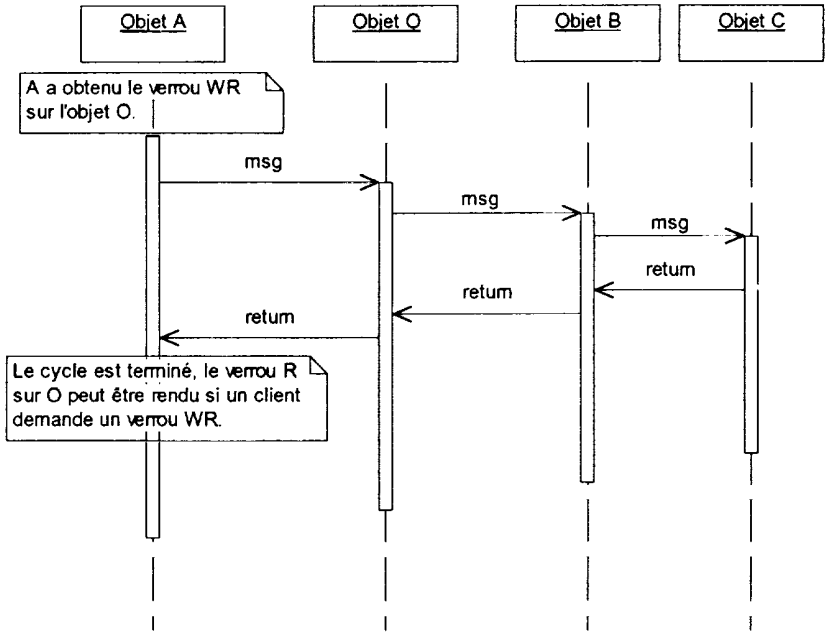


Fig. 214

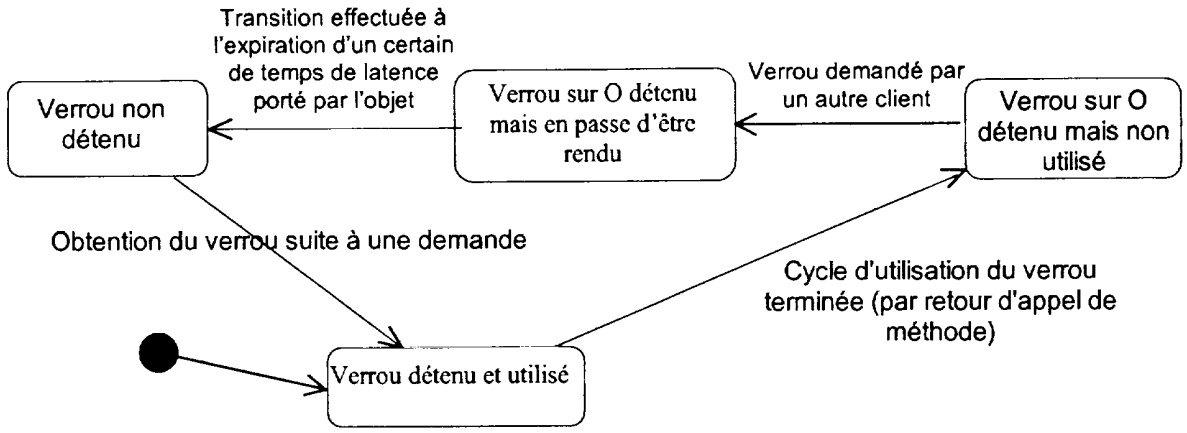


Fig. 215

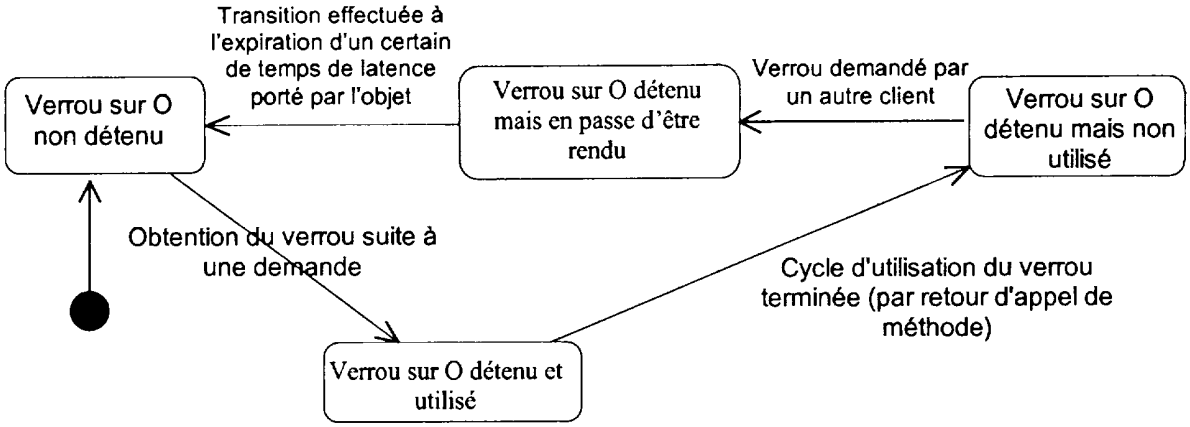


Fig. 216

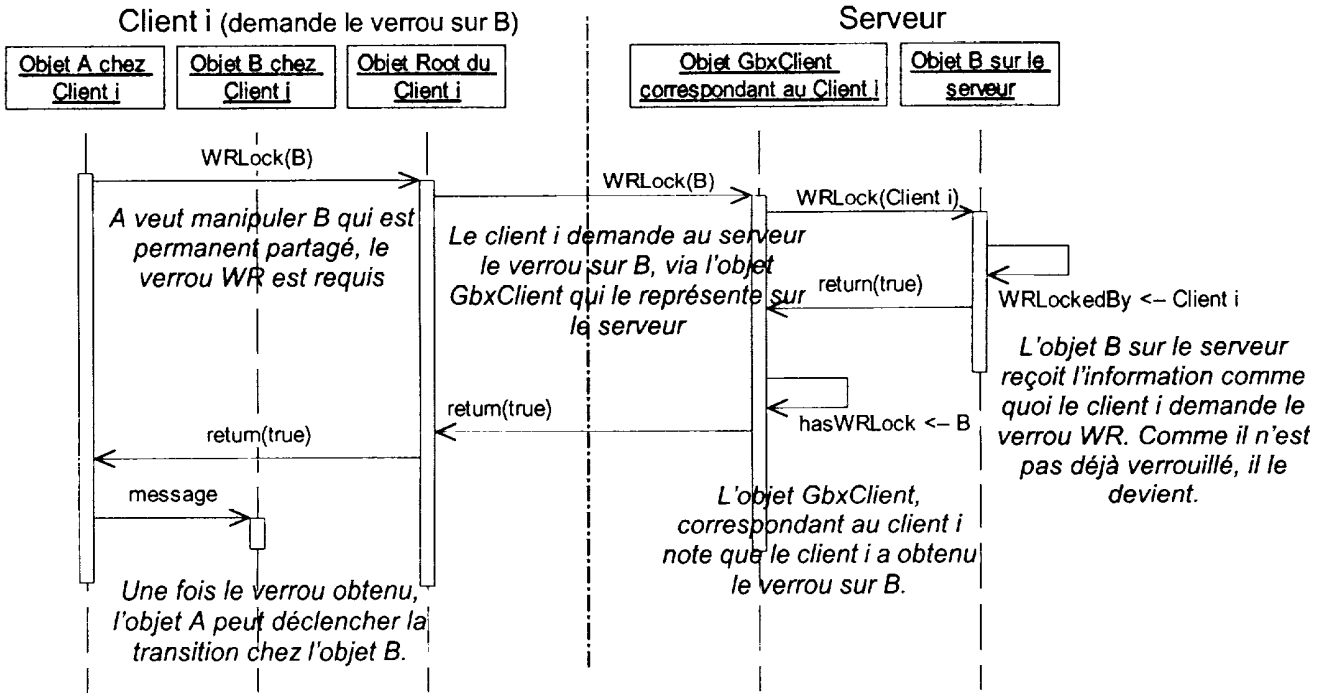


Fig. 217

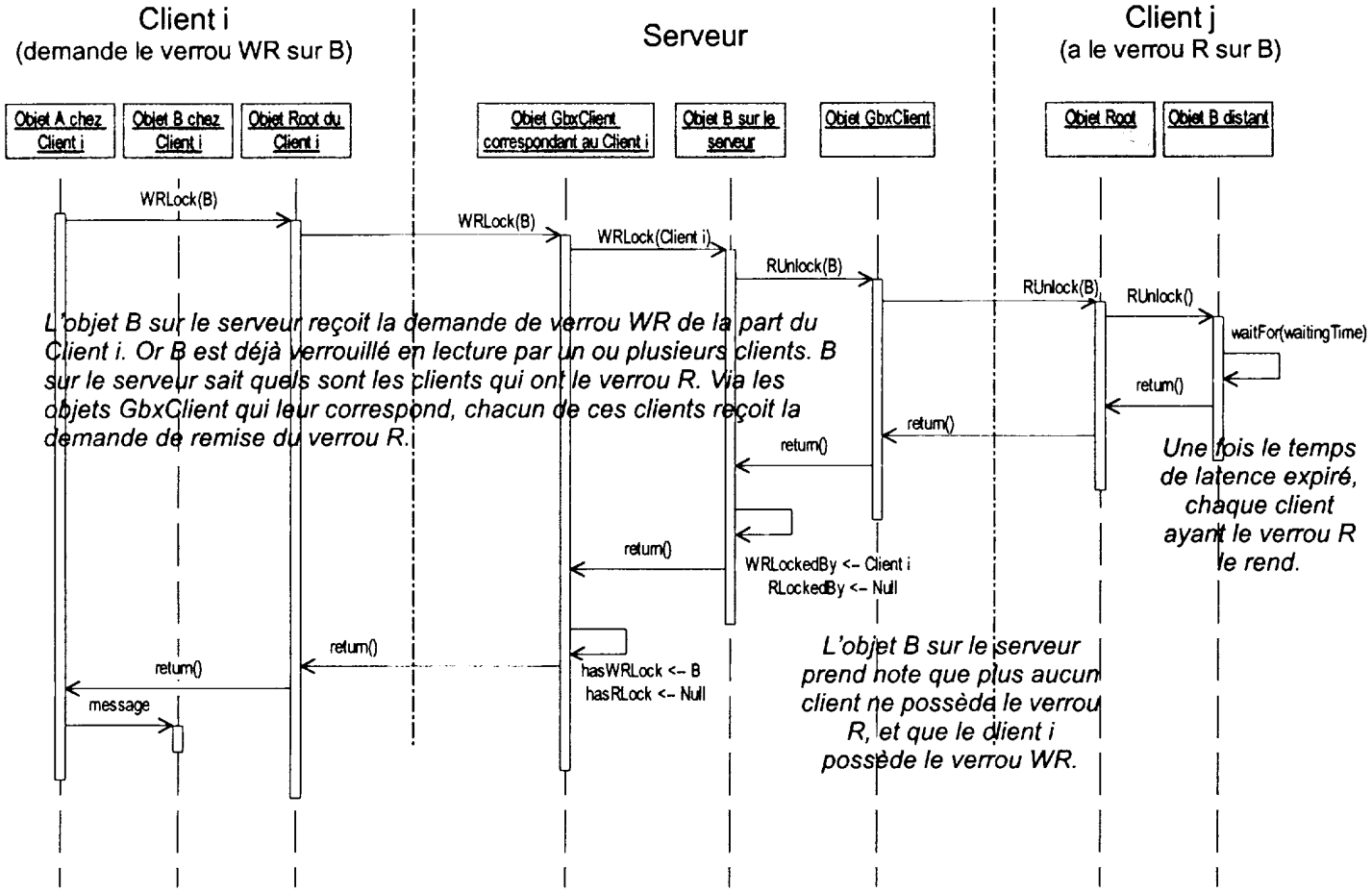

Client i
(demande le verrou WR sur B)
Serveur
Client j
(a le verrou R sur B)
Objet A chez Client i
Objet B chez Client i
Objet Root du Client i
Objet GbxClient correspondant au Client i
Objet B sur le serveur
Objet GbxClient
Objet Root
Objet B distant
WRLock(B)
WRLock(B)
WRLock(Client i)
RUnlock(B)
RUnlock(B)
RUnlock()
waitFor(waitingTime)
return()
return()
return()
return()
return()
message
WRLockedBy <- Client i
RLockedBy <- Null
hasWRLock <- B
hasRLock <- Null
L'objet B sur le serveur reçoit la demande de verrou WR de la part du Client i. Or B est déjà verrouillé en lecture par un ou plusieurs clients. B sur le serveur sait quels sont les clients qui ont le verrou R. Via les objets GbxClient qui leur correspond, chacun de ces clients reçoit la demande de remise du verrou R.
Une fois le temps de latence expiré, chaque client ayant le verrou R le rend.
L'objet B sur le serveur prend note que plus aucun client ne possède le verrou R, et que le client i possède le verrou WR.

Fig. 218

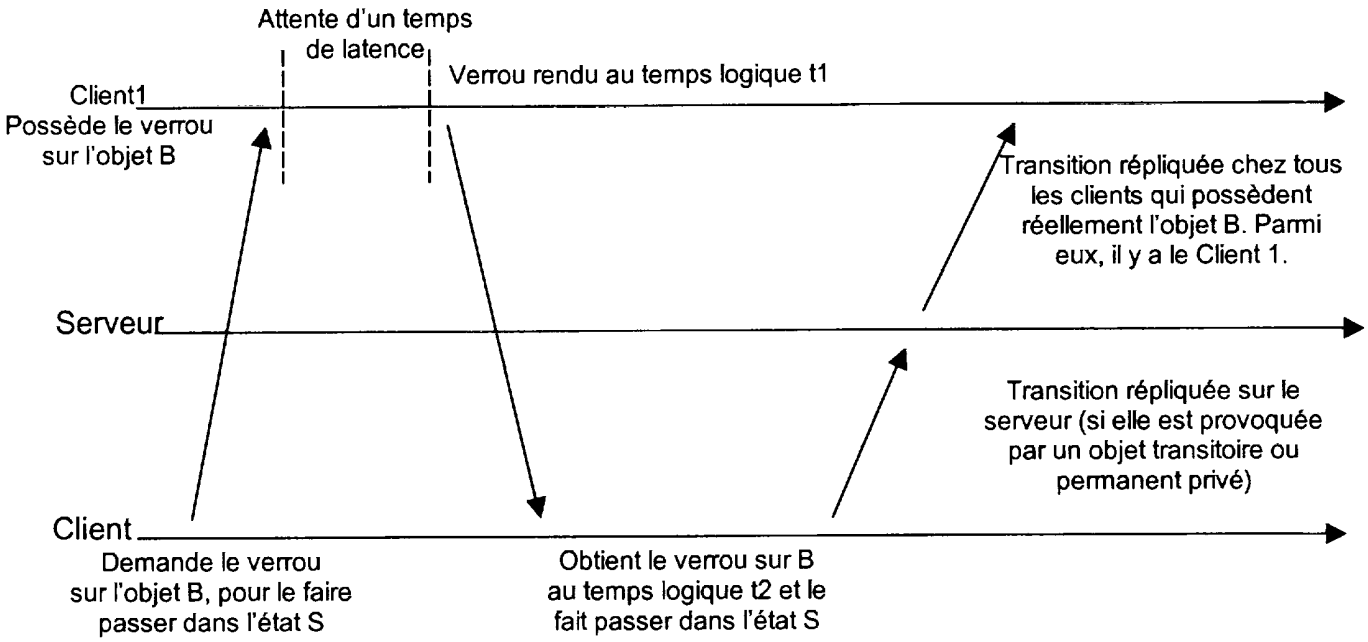


Fig. 219

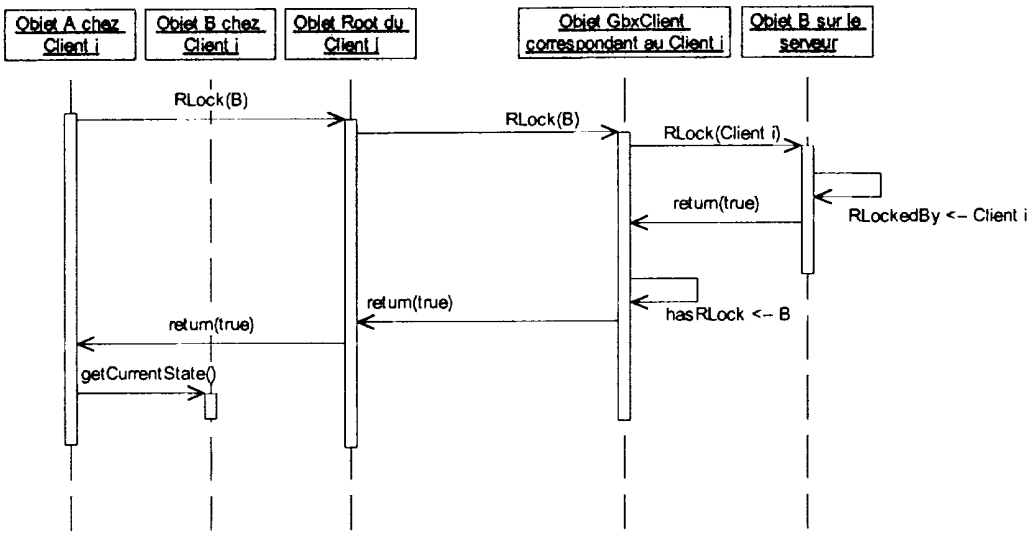


Fig. 220

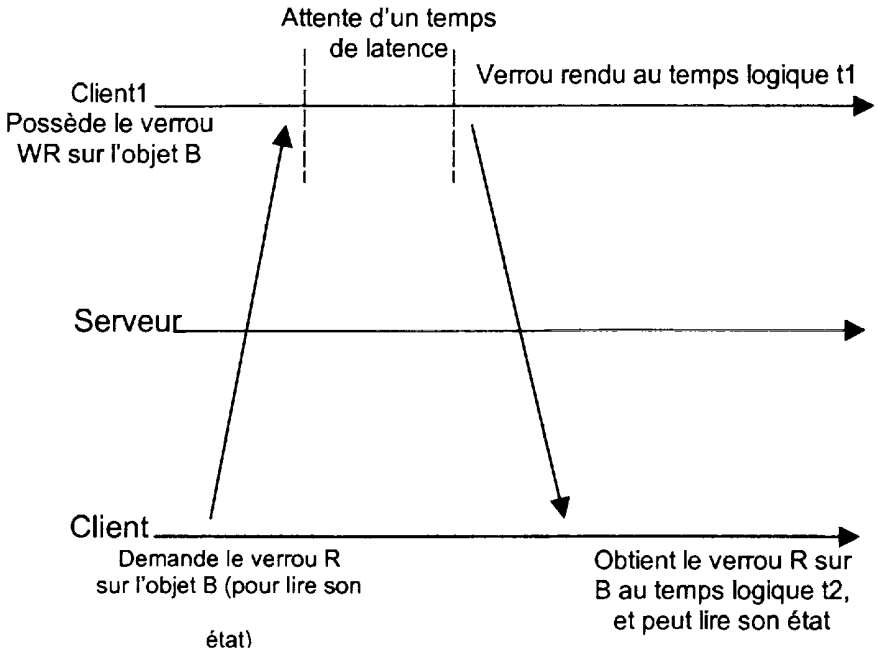

Attente d'un temps
de latence
Client1
Possède le verrou
WR sur l'objet B
Verrou rendu au temps logique t1
Serveur
Client
Demande le verrou R
sur l'objet B (pour lire son
état)
Obtient le verrou R sur
B au temps logique t2,
et peut lire son état

Fig. 221

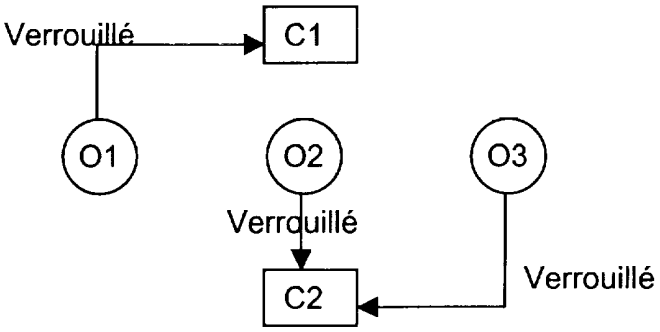


Fig. 222

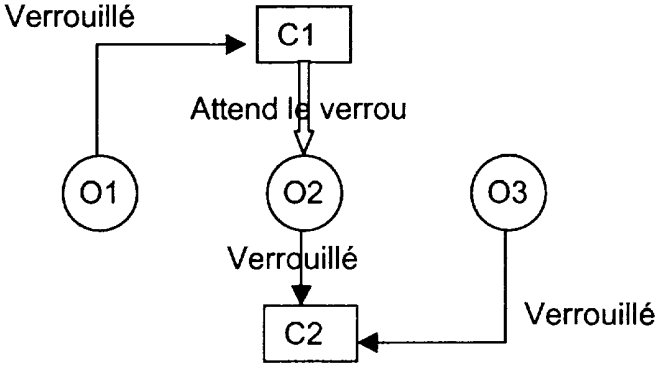


Fig. 223

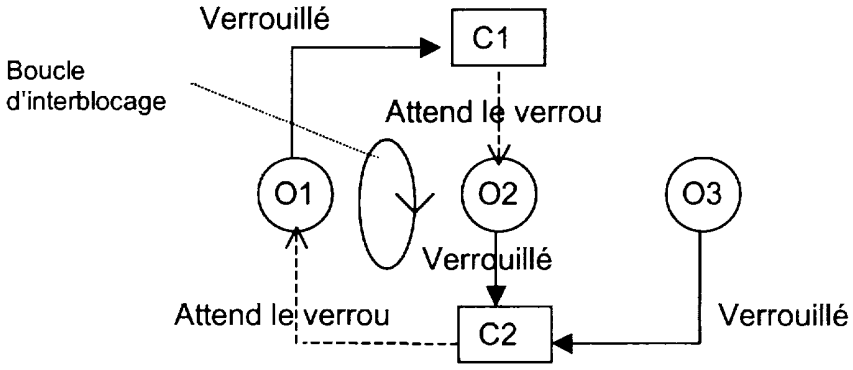


Fig. 224

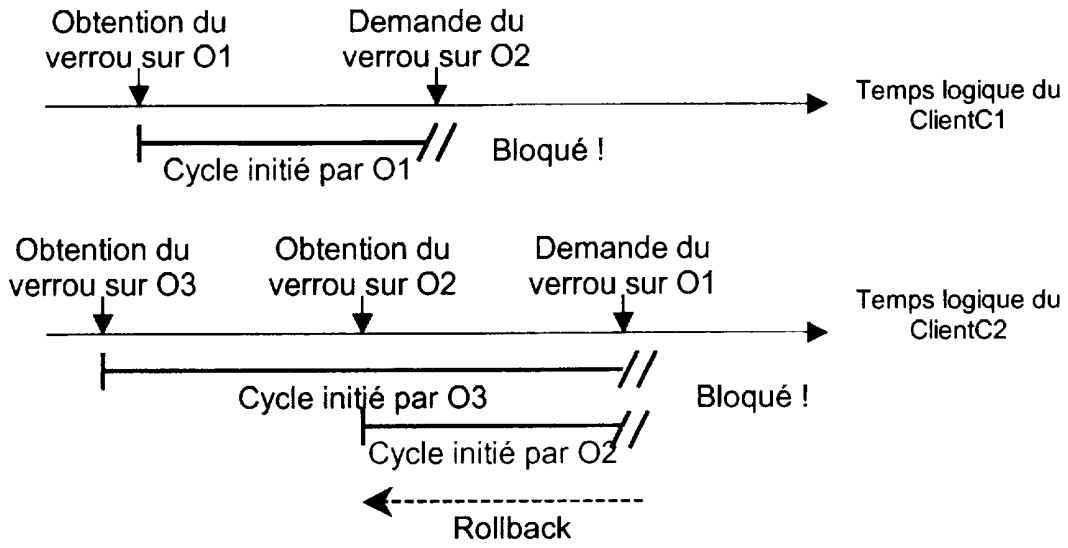
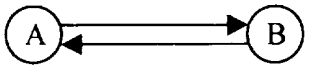
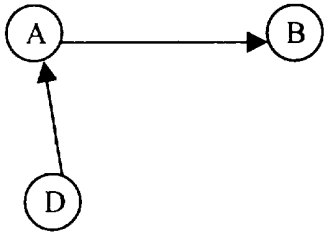


Fig. 225



A attend le verrou de B et reçoit une demande de verrou de la part de B : interblocage.



A attend le verrou de B et reçoit une demande de verrou de la part de D : pas d'interblocage.

Fig. 226

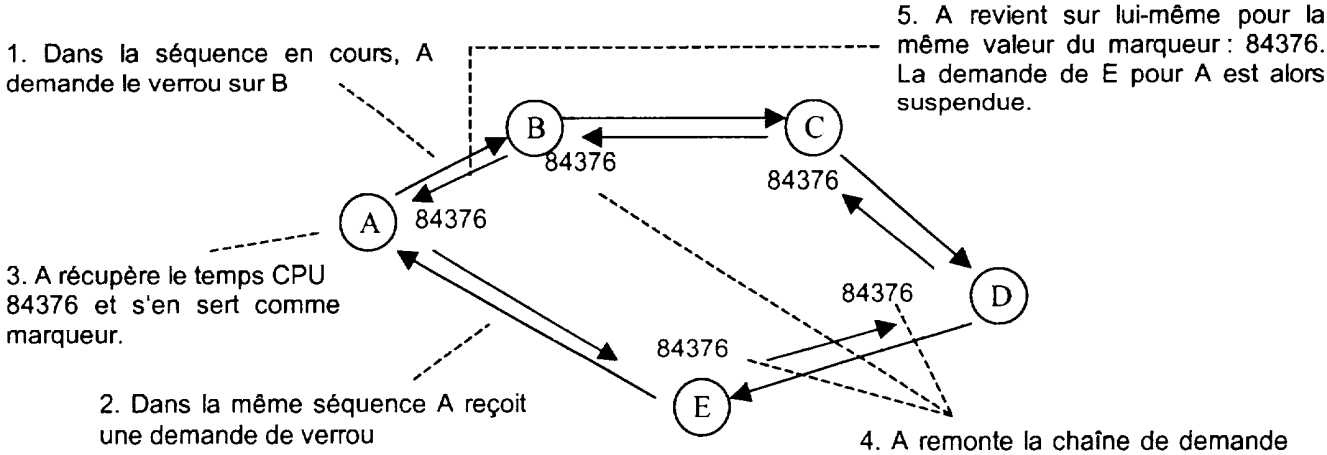


Fig. 227

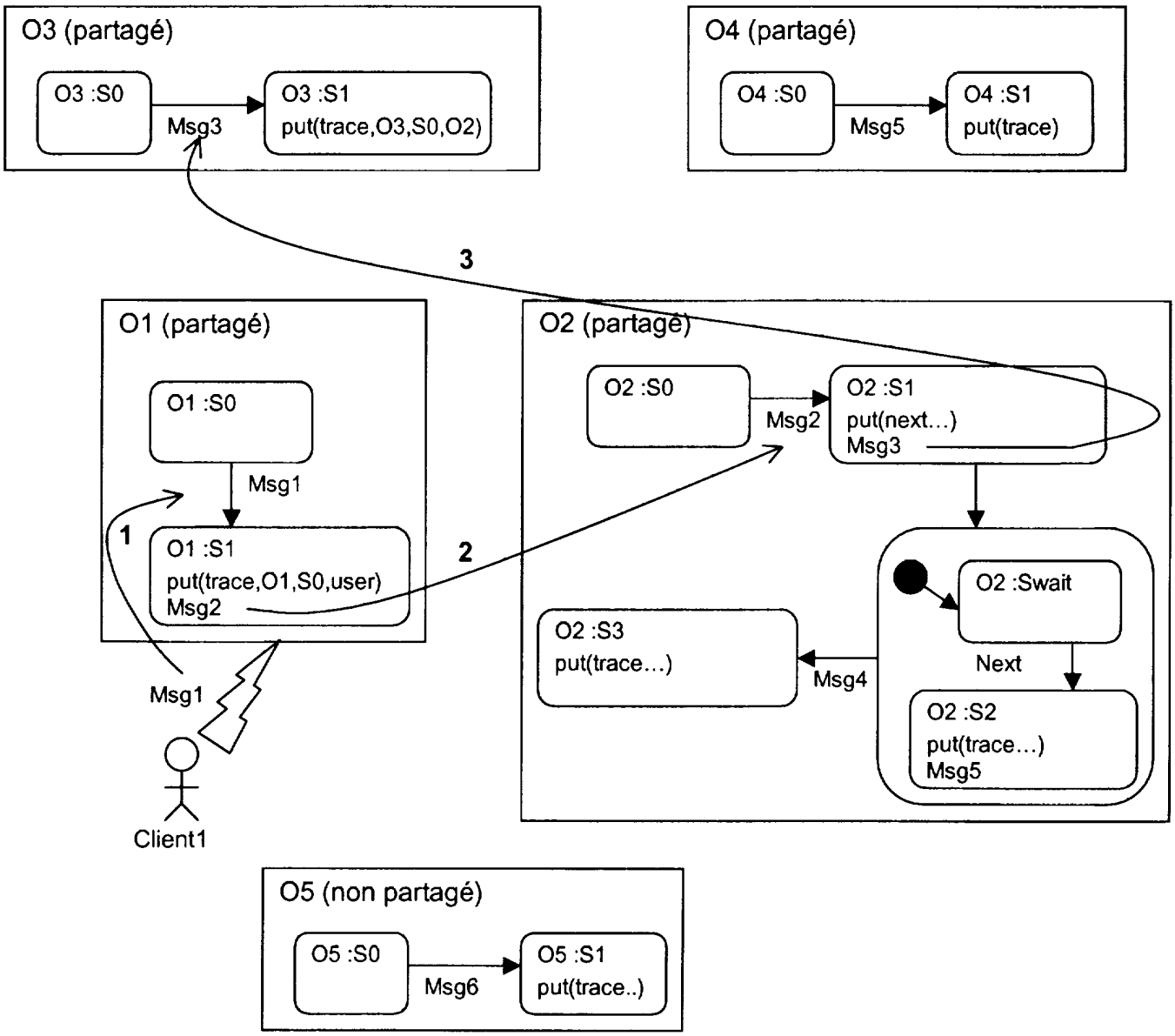


Fig. 228

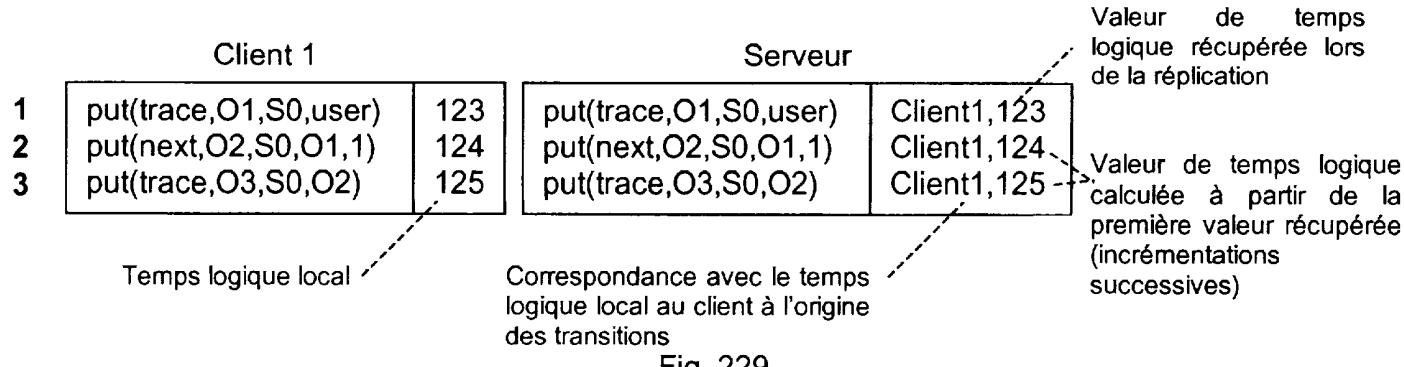


Fig. 229

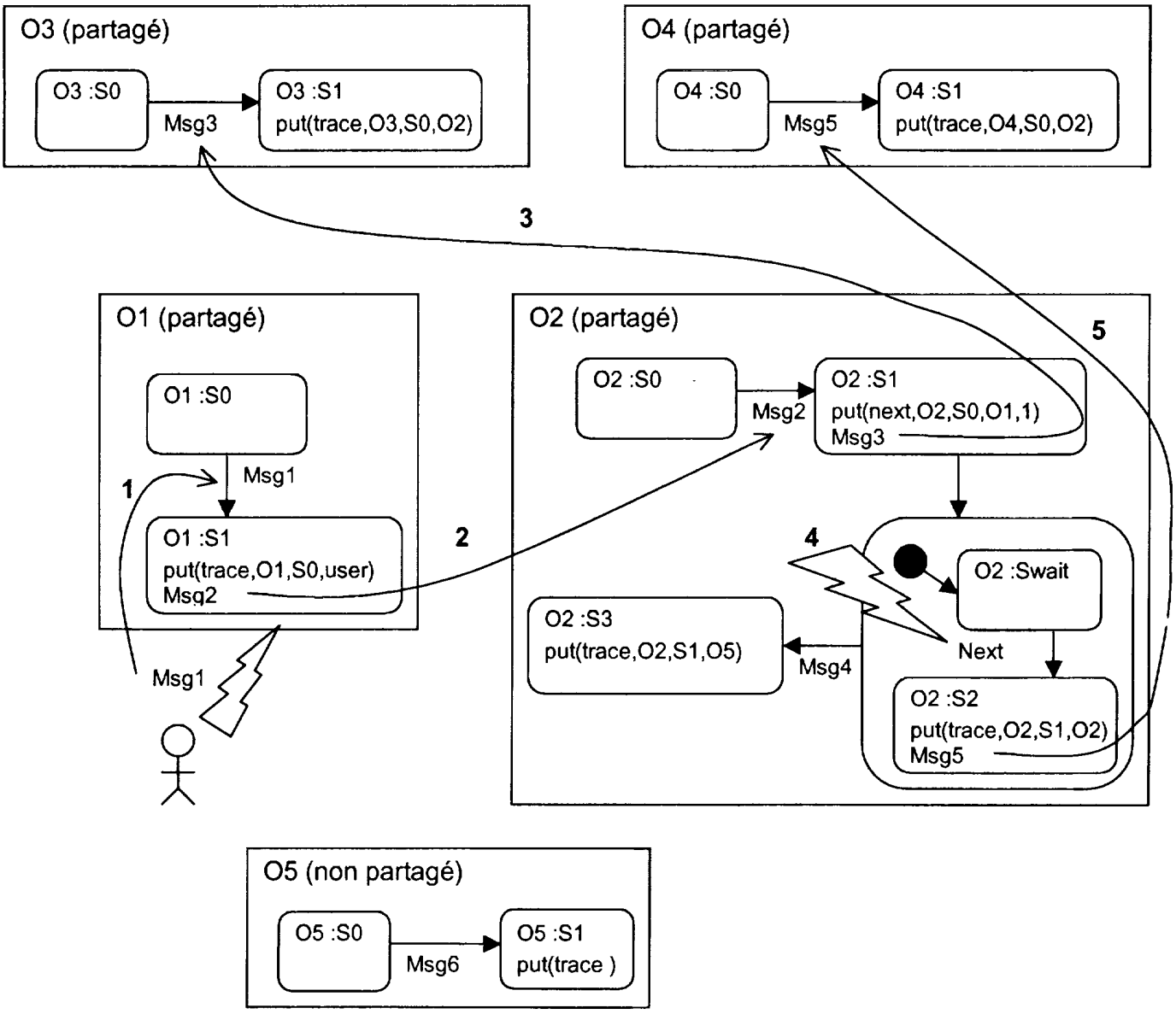


Fig. 230

Serveur

put(trace,O1,S0,user)	Client1,123
put(next,O2,S0,O1,1)	Client1,124
put(trace,O3,S0,O2)	Client1,125
put(trace,O2,S1,O2)	Client1,126
put(trace,O4,S0,O2)	Client1,127

Fig. 231

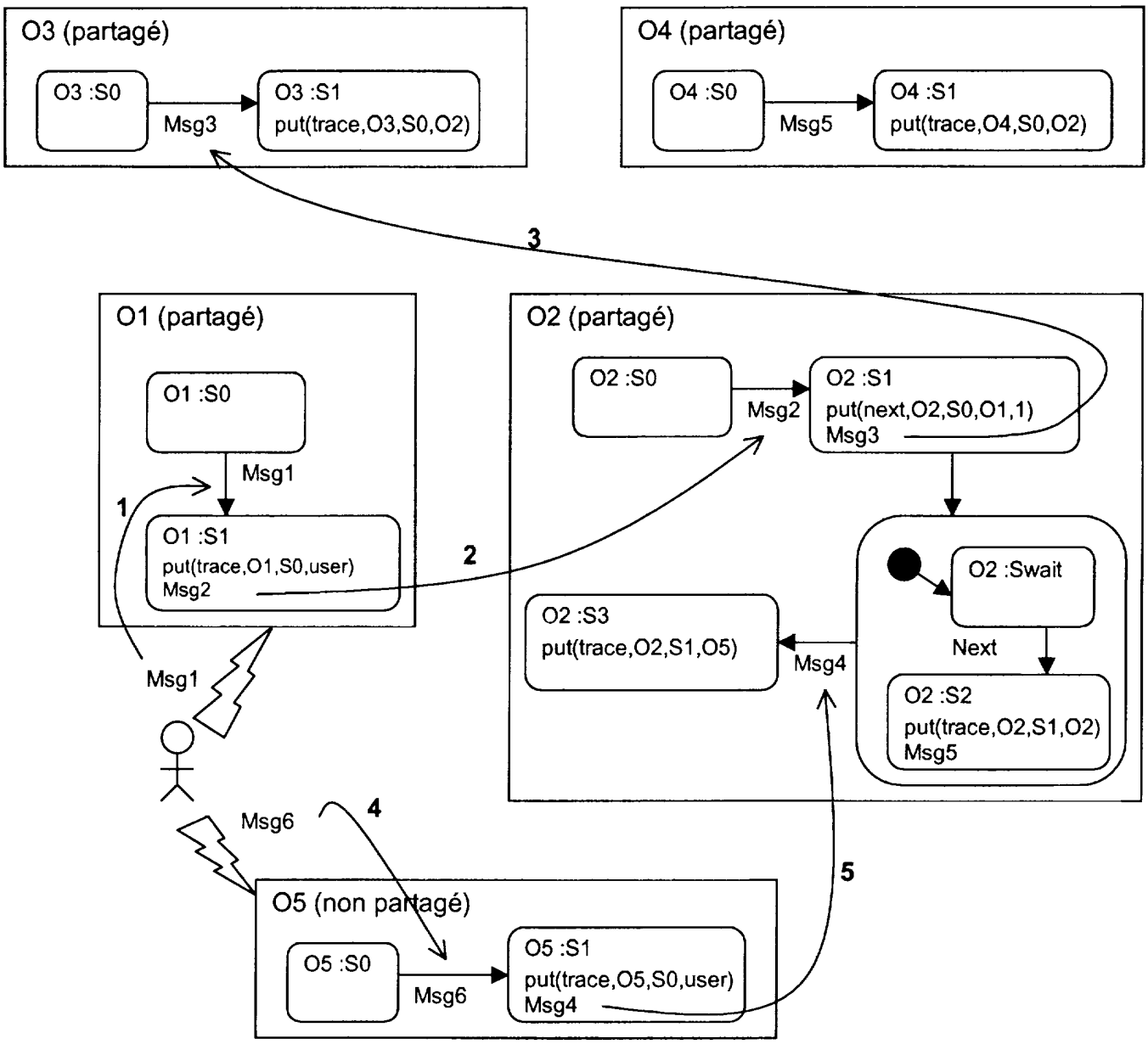


Fig. 232

Client1

put(trace,O1,S0,user)	123
put(next,O2,S0,O1,1)	124
put(trace,O3,S0,O2)	125
put(trace,O5,S0,user)	126
put(trace,O2,S1,O5)	127

Fig. 233

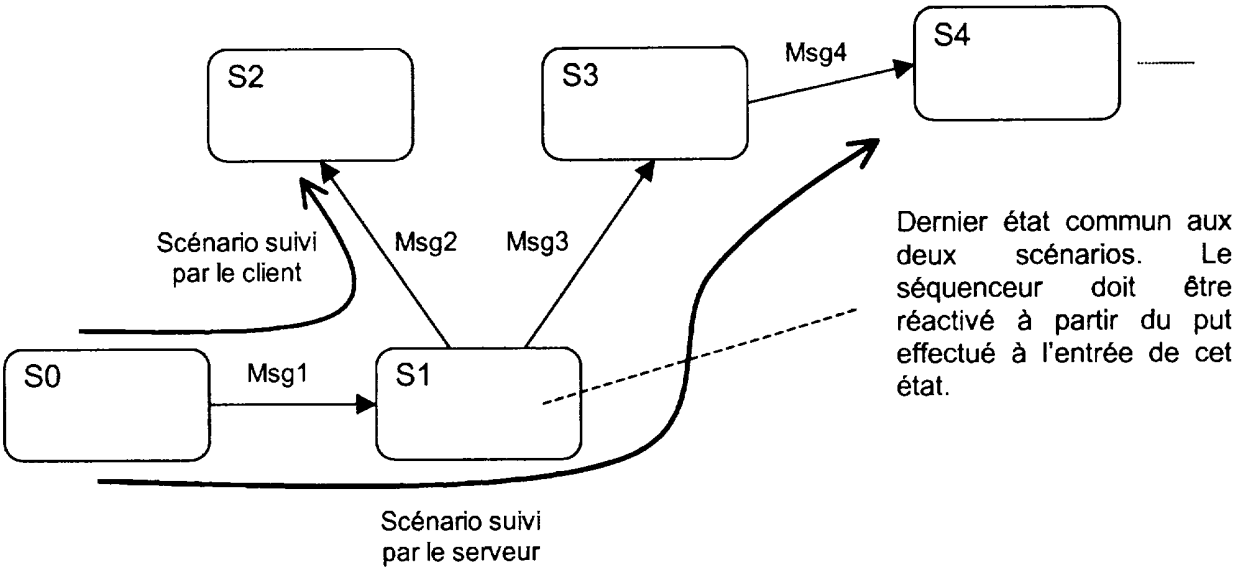


Fig. 234

2.

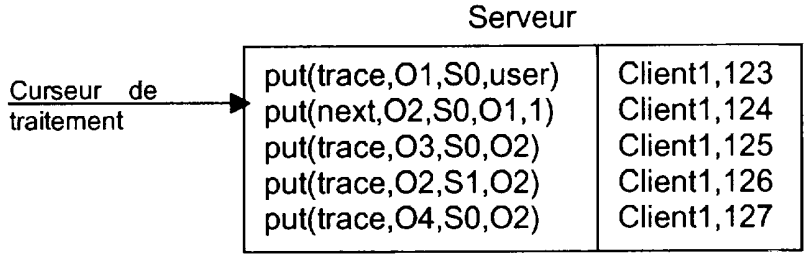


Fig. 235

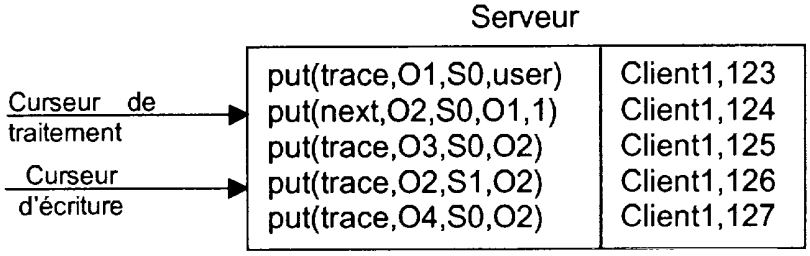


Fig. 236

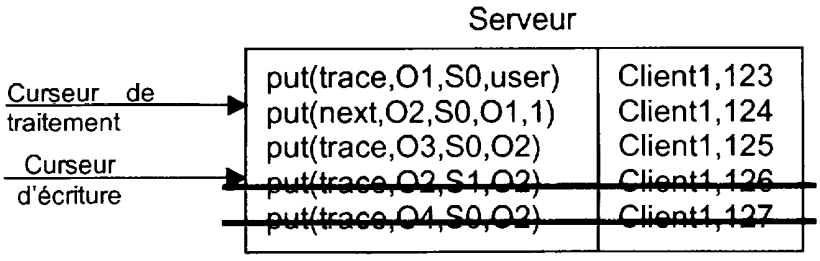


Fig. 237

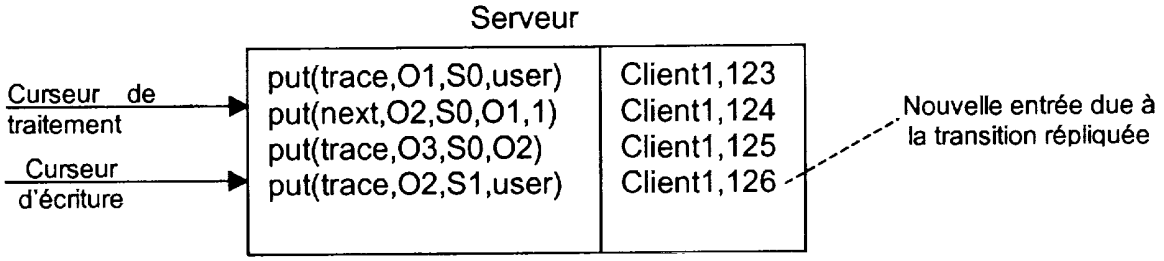

Serveur
Curseur de traitement
Curseur d'écriture
put(trace,O1,S0,user)
put(next,O2,S0,O1,1)
put(trace,O3,S0,O2)
put(trace,O2,S1,user)
Client1,123
Client1,124
Client1,125
Client1,126
Nouvelle entrée due à la transition répliquée

Fig. 238