

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION
EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété
Intellectuelle
Bureau international



(43) Date de la publication internationale
19 octobre 2006 (19.10.2006)

PCT

(10) Numéro de publication internationale
WO 2006/108865 A2

(51) Classification internationale des brevets : **Non classée**

(21) Numéro de la demande internationale :
PCT/EP2006/061567

(22) Date de dépôt international : 12 avril 2006 (12.04.2006)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
0503612 12 avril 2005 (12.04.2005) FR
0507016 1 juillet 2005 (01.07.2005) FR
0509001 2 septembre 2005 (02.09.2005) FR
0600213 10 janvier 2006 (10.01.2006) FR

AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO,
CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB,
GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG,
KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY,
MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO,
NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK,
SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ,
VC, VN, YU, ZA, ZM, ZW.

(84) États désignés (sauf indication contraire, pour tout titre
de protection régionale disponible) : ARIPO (BW, GH,
GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,
ZW), eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,
FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT,
RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA,
GN, GQ, GW, ML, MR, NE, SN, TD, TG).

(71) Déposant et

(72) Inventeur : **MAIM, Enrico** [FR/FR]; 8, Rue Dareau,
F-75014 Paris (FR).

(74) Mandataire : **LE FORESTIER, Eric**; Cabinet Regim-
beau, 20, rue de Chazelles, F-75847 Paris Cedex 17 (FR).

(81) États désignés (sauf indication contraire, pour tout titre de
protection nationale disponible) : AE, AG, AL, AM, AT,

Publiée :

— sans rapport de recherche internationale, sera republiée
dès réception de ce rapport

En ce qui concerne les codes à deux lettres et autres abrégia-
tions, se référer aux "Notes explicatives relatives aux codes et
abréviations" figurant au début de chaque numéro ordinaire de
la Gazette du PCT.

(54) Title: METHODS FOR GRANTING ACCESS TO RESOURCES MODIFIABLE BY USERS IN A COMPUTER ENVI-
RONMENT, AND RESOURCES STRUCTURED THEREFOR

(54) Titre : PROCEDES POUR PERMETTRE L'ACCES A DES RESSOURCES MODIFIABLES PAR DES UTILISATEURS
DANS UN ENVIRONNEMENT INFORMATIQUE, ET RESSOURCES STRUCTUREES A CET EFFET

(57) Abstract: The invention concerns a method for accessing a resource in a computer environment, the resource comprising a set of objects. The invention is characterized in that the computer environment is adapted to store in association with at least one object of said resource at least one modified object, in that said computer environment comprises means for storing in association with such an object information concerning degree of elaboration, and in that it includes the following steps: identifying, for each object of the resource to which corresponds at least one modified object, based on data concerning degree of elaboration, at least one most elaborated version of said object; assembling the objects thus identified for display in the resource. The invention also concerns other methods for accessing to resources as well as resources structured therefor.

(57) Abrégé : L'invention propose un procédé pour accéder à une ressource dans un environnement informatique, la ressource comprenant un ensemble d'objets. Il est remarquable en ce que l'environnement informatique est apte à mémoriser en association avec au moins un objet de ladite ressource au moins un objet modifié, en ce que ledit environnement informatique comprend des moyens aptes à mémoriser en association avec un tel objet une information de degré d'élaboration, et en ce qu'il comprend les étapes suivantes : identifier, pour chaque objet de la ressource auquel correspond au moins un objet modifié, à partir des informations de degré d'élaboration, au moins une version la plus élaborée dudit objet ; assembler les objets ainsi identifiés en vue de leur affichage dans la ressource. D'autres procédés pour accéder à des ressources sont également couvertes, ainsi que des ressources structurées à cet effet, sont également proposés.

WO 2006/108865 A2

Titre : « Procédés pour permettre l'accès à des ressources modifiables par des utilisateurs dans un environnement informatique, et ressources structurées à cet effet »

5 La présente invention concerne d'une façon générale le domaine de la collaboration entre utilisateurs dans un environnement informatique dans lequel des ressources, obtenues par assemblage d'objets, peuvent être accédées, dérivées, complétées, modifiées, etc. par ces utilisateurs.

Un exemple typique de ressource est une page Web accessible par l'Internet, mais la présente invention n'est nullement limitée à ce type de ressource.

10 On connaît déjà, notamment par des précédentes publications de brevets au nom du Demandeur, des techniques permettant d'accéder à des ressources, de les modifier, etc. de manière à permettre aux utilisateurs d'enrichir progressivement le contenu de ces ressources, et de construire des versions personnelles de celles-ci, auxquelles d'autres informations peuvent être ajoutées.

15 Le Demandeur a également proposé des techniques permettant de déterminer la pertinence de ressources en fonction de leur environnement en termes de ressources citantes et de ressources citées.

La présente invention a pour objet principal de proposer des procédés d'accès à des ressources, de transformation ou personnalisation de ressources en fonction de l'utilisateur, et de création
20 de nouvelles ressources, dans lesquels on puisse intégrer des informations de degré d'élaboration des ressources, de manière à proposer par priorité aux utilisateurs des informations a priori les plus élaborées, par opposition à d'autres informations moins élaborées. Un autre objet de l'invention est de déterminer ainsi quels objets d'information doivent être remplacés ou enrichies (par des objets alternatifs) lors de la présentation d'une ressource à
25 l'utilisateur.

L'invention propose ainsi de nouveaux moyens de synchronisation et enrichissement d'informations couvrant tous formats, aussi bien de texte et images que le multimédia, et notamment les flux RSS et les « podcasts » (audio, vidéo).

30 Un autre objet, corollaire, de la présente invention est de proposer des procédés d'accès à des ressources, de transformation ou personnalisation de ressources en fonction de l'utilisateur, et de création de nouvelles ressources, dans lesquelles les propriétaires des ressources d'origine puissent avoir une influence, notamment en termes de droit de modification et de diffusion, sur les ressources qui en sont dérivées.

35 Ainsi la présente invention propose selon un premier aspect un procédé pour accéder à une ressource dans un environnement informatique, la ressource comprenant un ensemble d'objets, caractérisé en ce que l'environnement informatique est apte à mémoriser en association avec au moins un objet de ladite ressource au moins un objet modifié, en ce que ledit environnement informatique comprend des moyens aptes à mémoriser en association avec un tel objet une information de degré d'élaboration, et en ce que le procédé comprend les étapes
40 suivantes :

- identifier, pour chaque objet de la ressource auquel correspond au moins un objet modifié, à partir des informations de degré d'élaboration, au moins une version la plus élaborée dudit objet,

- assembler les objets ainsi identifiés en vue de leur affichage dans la ressource.

45 Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

- * les informations de degré d'élaboration d'un objet découlent de la chronologie des modifications de l'objet, en ce qu'il est prévu lors de la modification d'un objet par un utilisateur une étape permettant à d'autres utilisateurs de refuser l'objet modifié, et en ce que l'étape d'assemblage est réalisée avec une version de l'objet choisie en fonction de l'existence
50 ou non de refus.

* le procédé comprend, lorsqu'un lification d'un objet est refusée, la création de deux branches contenant deux versions alternatives de l'objet, à savoir la version modifiée et la version non modifiée.

5 * l'étape d'assemblage d'une ressource incluant une certaine version de l'objet en question est effectuée en signalant l'existence d'une version ou de versions alternatives dudit objet.

* le procédé comprend en outre une étape de sélection, par un utilisateur accédant à une version de la ressource, d'une certaine version alternative de l'objet.

10 * des versions plus élaborées des objets sont formées indépendamment pour les différents objets alternatifs dans chacune des branches.

* le procédé comprend en outre la mise en œuvre de moyens de calcul de scores de pertinences des différentes versions alternatives des objets pour présenter sélectivement les versions les plus pertinentes.

15 * plusieurs versions d'une ressource peuvent être engendrées par des utilisateurs, et en ce que dans l'étape d'assemblage, on propose par défaut la version de l'objet en fonction de la version de la ressource.

* il est prévu une étape d'inhibition de l'utilisation d'un objet dans l'assemblage d'une ressource si une condition donnée relative à cet objet ou à un autre objet de la ressource à assembler n'est pas remplie.

20 * le contenu d'un objet dont l'utilisation peut être inhibée est crypté, et en ce que l'étape d'inhibition comprend l'inaccessibilité à une clé de décryptage.

* la condition donnée est une condition de présence d'au moins un objet d'une autre ressource.

* la condition donnée est une condition de non-modification de l'objet présent.

25 * le procédé est mis en œuvre dans un environnement informatique comprenant :

- au moins un serveur de contenus apte à fournir des contenus d'objets pouvant être assemblés pour former des ressources,

- un gestionnaire de collaboration apte à gérer les versions des ressources et les versions des objets qu'elles contiennent,

30 - au niveau d'au moins un poste utilisateur, une extension (plug-in) d'un outil de manipulation d'objets et de ressources, apte à communiquer avec le gestionnaire de collaboration,

et en ce qu'il comprend une étape de vérification de l'authenticité de l'extension par le gestionnaire de collaboration.

35 * l'information de degré d'élaboration est apte à prendre l'une parmi deux valeurs, à savoir « le plus élaboré » ou non.

* l'étape d'identification comprend la recherche d'objets (OR ; OA) dont le degré d'élaboration possède la valeur « le plus élaboré » parmi les objets aval successivement modifiés à partir de l'objet en question de la ressource accédée.

40 * l'étape d'identification comprend la recherche d'objets de degré d'élaboration supérieur à une limite parmi les objets aval successivement modifiés à partir de l'objet en question de la ressource accédée.

* la recherche inclut la recherche d'objets de degré(s) d'élaboration le(s) plus élevé(s) en parcourant les objets amont par modification desquels l'objet en question a été obtenu.

45 * dans le cas où l'étape d'identification a permis d'identifier plusieurs objets les plus élaborés, il est prévu dans l'étape d'assemblage des objets une sous-étape d'ajout à la ressource d'une information signalant l'existence de tels plusieurs objets (OA).

* l'objet sélectionné pour l'assemblage dans la ressource est l'objet aval le plus élaboré le plus proche dans la succession.

50 * en réponse à des instructions de changement d'objet reçues de l'utilisateur, l'étape d'assemblage est apte à utiliser successivement pour l'assemblage des objets les plus élaborés triés selon un autre critère tel qu'une notation.

* le procédé est apte, en réponse à une instruction de sélection par un utilisateur, à utiliser pour l'assemblage un objet désigné par ladite instruction.

* le procédé comprend en outre une étape d'ajustement de la notation de l'objet en fonction d'une instruction de sélection de cet objet par l'utilisateur.

5 * l'étape d'assemblage est apte à utiliser pour l'assemblage un objet sélectionné selon un autre critère tel qu'une notation parmi tous les objets les plus élaborés.

10 * le procédé est mis en œuvre en combinaison avec un serveur de contenus préexistants doté de moyens de gestion chronologique de versions des objets et en ce que l'étape d'identification de la version la plus élaborée d'un objet est mise en œuvre à partir d'informations de versions fournis par lesdits moyens de gestion chronologique et d'informations de contribution d'un utilisateur accédant à la ressource aux différentes versions, et en ce qu'il comprend en outre une étape de présentation de la version courante de l'objet dans le serveur de contenus.

15 Selon un deuxième aspect, on propose un procédé mis en œuvre en association avec des moyens de mémorisation accessibles dans un environnement informatique pour maintenir à jour des graphes d'objets susceptibles d'être assemblés pour former des ressources accessibles dans l'environnement, caractérisé en ce que les moyens de mémorisation sont aptes à mémoriser en association avec ces objets une information de degré d'élaboration, et en ce que
20 le procédé comprend les étapes suivantes :

- détecter la modification d'un objet par un utilisateur,
- identifier des propriétaires de ressources contenant des objets correspondants à l'amont,
- diriger vers ces propriétaires une proposition d'acceptation ou de refus de cette
25 modification,
- attendre une réponse des propriétaires,
- en fonction du contenu d'au moins la première réponse, ajuster les informations de degré d'élaboration des objets.

30 Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

* l'information de degré d'élaboration est apte à prendre l'une parmi deux valeurs, à savoir « le plus élaboré » ou non.

35 * le procédé comprend en outre une étape consistant, en réponse à la détection de la modification d'un objet, à ajuster temporairement les informations de degré d'élaboration des objets, cet ajustement pouvant être contrarié ou confirmé à l'issue d'au moins la première réponse des propriétaires.

* lors d'une première réponse d'acceptation d'un propriétaire, on applique la modification dans la ressource dudit propriétaire.

40 * lors d'une réponse de refus d'un propriétaire, alors l'étape d'ajustement donne à l'objet immédiatement en amont de l'objet modifié un degré d'élaboration le plus élevé.

* le procédé comprend en outre, lors d'une réponse de refus, l'ajout au graphe d'une branche contenant une version alternative de l'objet constituée par l'objet incluant la modification.

45 * le procédé comprend en outre, lors d'une réponse de refus d'une modification effectuée sur un objet qui n'est pas le plus élaboré et en cas d'incompatibilité de la modification avec au moins objet plus élaboré, l'ajout au graphe d'une branche contenant une version alternative de l'objet constituée par l'objet incluant la modification ou contenant le ou les objets incompatibles et les objets modifiés à partir d'eux.

50 Selon un troisième aspect de l'invention, on propose un procédé de création d'une nouvelle ressource à partir d'une ressource existante à laquelle on a accédé dans un environnement informatique, caractérisé en ce qu'il comprend les étapes suivantes :

- 4
- détecter la modification d'un objet existante par un utilisateur,
 - en réponse à cette détection, créer une nouvelle ressource présentant le même contenu que la ressource existante,
 - appliquer les modifications à l'objet de ladite autre ressource qui correspond à l'objet
- 5 de la ressource existante pour lequel la modification a été détectée.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

- * l'étape de création d'une nouvelle ressource est effectuée par transclusion à partir d'informations associées à la ressource existante.
- 10 * le procédé est mis en œuvre en association avec des moyens de mémorisation accessibles dans un environnement informatique pour maintenir à jour des graphes d'objets susceptibles d'être assemblés pour former des ressources accessibles dans l'environnement, les moyens de mémorisation sont aptes à mémoriser en association avec chacun de ces objets une information de degré d'élaboration, et le procédé comprend les étapes suivantes :
- 15 - détecter la modification d'un objet par un utilisateur,
 - en réponse à cette détection, créer une nouvelle ressource présentant le même contenu que la ressource existante,
 - appliquer les modifications à l'objet de ladite autre ressource qui correspond à l'objet de la ressource existante pour lequel la modification a été détectée,
 - 20 - identifier des propriétaires de ressources contenant des objets correspondants à l'amont,
 - diriger vers ces propriétaires une proposition d'acceptation ou de refus de cette modification,
 - attendre une réponse des propriétaires,
 - 25 - en fonction du contenu d'au moins la première réponse, ajuster les informations de degré d'élaboration des objets.

- Selon un quatrième aspect, on propose un procédé pour propager des informations standard de description d'objet entre différents objets susceptibles d'être assemblés pour former des
- 30 ressources visualisables dans une interface utilisateur au sein d'un environnement informatique, caractérisé en ce qu'à chaque objet sont associées des méta-données de description, et en ce que le procédé comprend les étapes suivantes :
- l'entrée dans un mode de propagation à l'initiative de l'utilisateur,
 - la sélection d'un objet dont on souhaite emprunter les méta-données de description,
 - 35 - la sélection d'au moins un autre objet pour lui affecter les mêmes méta-données de description.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

- * l'interface utilisateur est adaptée pour masquer à l'utilisateur les méta-données de
- 40 description.

- On propose selon un cinquième aspect de l'invention un procédé pour rendre accessible à des tiers au sein d'un environnement informatique une ressource contenant au moins un objet dérivé d'une autre ressource, caractérisé en ce qu'il comprend les étapes suivantes :
- 45 - détecter une requête d'un utilisateur en vue de modifier l'objet,
 - en réponse à cette détection, engendrer dans l'environnement informatique une requête auprès d'un propriétaire de ladite autre ressource en vue d'autoriser l'accessibilité de l'objet modifié,
 - en fonction de la réponse reçue du propriétaire, sélectivement autoriser ou non
- 50 l'accessibilité.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

* une requête commune est engendrée par une pluralité d'objets dérivés de ressources appartenant au même propriétaire.

On propose selon un sixième aspect de l'invention une ressource informatique apte à permettre la mise en œuvre du procédé selon le cinquième aspect précité, comprenant un ensemble d'objets et accessible au travers d'un environnement informatique, caractérisée en ce qu'en association avec au moins certains des objets est mémorisée une méta-donnée de contrainte de diffusion du contenu de l'objet, permettant de sélectivement contrôler sa modification et/ou son accessibilité par des tiers au sein de ressources dérivées.

Certains aspects préférés mais non limitatifs de cette ressource sont les suivants :

* les méta-données de contraintes de diffusion comprennent au moins une méta-donnée choisie dans le groupe comprenant les méta-données d'autorisation/interdiction de modification, des méta-données d'autorisation/interdiction d'accès aux tiers des objets non modifiés ou modifiés, et des méta-données de contact d'une autorité décisionnaire.

* la ressource est structurée sous forme d'une arborescence d'objets, en ce qu'en association avec au moins certains nœuds de l'arborescence, une méta-donnée de contrainte de diffusion est mémorisée, et en ce qu'une méta-donnée de contrainte de diffusion au niveau d'un nœud s'applique aux nœuds enfants du nœud en question.

Selon un septième aspect de l'invention est proposé un procédé pour rendre accessible à des tiers au sein d'un environnement informatique une ressource contenant au moins un objet dérivé d'une autre ressource, caractérisé en ce qu'aux objets sont susceptibles d'être associées, de façon mémorisée dans l'environnement informatique, des méta-données de contrainte de diffusion et ces méta-données étant susceptibles de contenir une identification d'un propriétaire d'un objet à partir duquel l'objet précité doit être dérivé, et en ce que le procédé comprend les étapes suivantes :

- détection du fait qu'un objet contenu dans une ressource à laquelle un utilisateur souhaite accéder est dérivée d'une autre ressource,

- détection de l'existence dans l'environnement informatique de méta-données de contrainte de diffusion de l'objet en question,

- dans l'affirmative, lecture desdites méta-données,

- détection de l'existence dans lesdites méta-données d'une identification d'une autorité décisionnaire,

- dans l'affirmative, envoi à ladite autorité ainsi identifiée d'une requête d'autorisation d'accessibilité à des tiers, et

- dans le cas où une acceptation de ladite requête est reçue, mise à disposition de la ressource auxdits tiers.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

* la mise à disposition de la ressource est une publication.

* le procédé comprend l'étape intermédiaire consistant, entre l'envoi de la requête et la réception d'une autorisation ou d'un refus, une étape d'assemblage provisoire de la ressource avec un objet formé d'un extrait de l'objet que l'on souhaite dériver.

* lorsque l'objet est un objet texte, ledit extrait est élaboré en fonction de critères juridiques de droit de citation.

Selon un huitième aspect, on propose selon l'invention un procédé pour estimer quantitativement l'activité d'objets dans un environnement informatique, cet environnement permettant l'accès à des ressources réalisées par assemblage d'objets, au moins certains des objets pouvant être engendrés par dérivation d'objets préexistants et modification des objets ainsi dérivés, procédé caractérisé en ce qu'il comprend les étapes suivantes :

- 6
- identifier des objets les plus ar à être dérivés, de telle sorte que cette identification se propage aux objets dérivés de ceux-ci,
 - compter les nombres de dérivation des objets les plus amont et des objets qui en ont été dérivés.

5

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

- * le procédé comprend en outre une étape de comptage groupé des consultations des objets les plus amont et des objets qui en ont été dérivés.

- * les objets les plus amont sont des objets à caractère publicitaire.

10

- * les objets les plus amont sont aptes à être dérivés sans pouvoir être modifiés.

15

Un neuvième aspect de l'invention consiste en un environnement informatique pour mettre en œuvre un procédé tel que défini dans ce qui précède, caractérisé en ce qu'il comprend une pluralité de serveurs reliés en maillage, aptes à contenir des informations de dérivation entre objets et de marquage de degrés d'élaboration, chaque serveur étant apte à recevoir une requête en vue d'indiquer des identifiants d'objets sur la base desdites informations, et étant apte à rediriger une telle requête vers un autre serveur dans le cas où il n'est pas apte à répondre à ladite requête.

20

Selon une première forme d'un dixième aspect de l'invention est proposé un procédé pour gérer des ressources à contenus variables dans un environnement informatique, chaque ressource comprenant un ensemble d'objets et au moins certains de ces objets étant présentés dans une ressource en formant un objet transclus individuellement ou en tant que partie d'un objet transclus plus grand à partir d'un objet source, via le cas échéant un ou plusieurs objets transclus intermédiaires, de telle sorte qu'une modification apportée à l'objet source puisse se propager jusqu'audit objet transclus, le procédé étant caractérisé :

25

- en ce que l'environnement informatique est apte à mémoriser des informations de transclusion basées sur des identifiants d'objets, informations à partir desquelles des objets sources ont la possibilité d'être présentés dans des objets transclus, et

30

- en ce que le procédé comprend l'étape consistant, lors de la modification d'un objet, à engendrer ou à mettre à jour, pour chaque objet transclus à partir dudit objet, un identifiant d'objet source à prendre en compte pour cet objet transclus, en fonction d'une information de distance entre l'objet modifié et ledit objet transclus.

35

Une deuxième forme de ce dixième aspect consiste en un procédé pour gérer des ressources à contenus variables dans un environnement informatique, chaque ressource comprenant un ensemble d'objets et au moins certains de ces objets étant présentés dans une ressource en formant un objet transclus individuellement ou en tant que partie d'un objet transclus plus grand à partir d'un objet source, via le cas échéant un ou plusieurs objets transclus intermédiaires, de telle sorte qu'une modification apportée à l'objet source puisse se propager jusqu'audit objet transclus, le procédé étant caractérisé :

40

- en ce que l'environnement informatique est apte à mémoriser des informations de transclusion basées sur des identifiants d'objets, informations à partir desquelles des objets sources ont la possibilité d'être présentés dans des objets transclus, et

45

- en ce que le procédé comprend l'étape consistant, lors de la présentation d'un objet, à déterminer, pour chaque objet transclus à partir dudit objet, un identifiant d'objet source à prendre en compte pour cet objet transclus, en fonction d'une information de distance entre l'objet modifié et ledit objet transclus.

50

Certains aspects préférés mais non limitatifs de ces procédés sont les suivants :

- * ladite étape, lors de la modification d'un objet, identifie cet objet comme sa propre source, avec une distance nulle.

* les ressources comprennent d arborescentes d'objets, et en ce qu'il comprend en outre une étape consistant, pour une transclusion d'un objet englobant des objets descendants dont l'un est transclus à partir de l'autre, à définir pour la transclusion entre objets descendants de l'objet englobant transclus, une valeur de distance entre ces objets plus courte.

5 * lesdits modifications peuvent concerner des attributs différents desdits objets, et en ce que l'étape de génération ou de mise à jour d'objet source est effectuée séparément pour chacun des attributs, un objet pouvant ainsi avoir des objets sources différents selon l'attribut considéré.

10 * les attributs différents comprennent au moins deux attributs parmi des attributs de contenu, de méta-données, de position et de transformation.

Selon un onzième aspect de l'invention, on prévoit un procédé pour gérer des ressources à contenus variables dans un environnement informatique, chaque ressource comprenant un ensemble d'objets et au moins certains de ces objets étant présentés dans une ressource en formant un objet transclus individuellement ou en tant que partie d'un objet transclus plus grand à partir d'un objet source, via le cas échéant un ou plusieurs objets transclus intermédiaires, de telle sorte qu'une modification apportée à l'objet source puisse se propager jusqu'audit objet transclus, caractérisé en ce qu'il comprend, selon la fréquence des modifications apportées aux objets et des présentations d'objets, la mise en œuvre soit de la première forme du dixième aspect de l'invention, soit de la deuxième forme dudit dixième aspect.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

25 * la sélection du procédé selon la première forme du dixième aspect est automatiquement sélectionnée quand l'environnement de l'objet source n'est pas apte à engendrer ledit identifiant d'objet source.

* la sélection du procédé mis en œuvre est effectuée transclusion par transclusion.

* le procédé est mis en œuvre dans des moyens serveurs centralisés aptes à fournir les identifiants d'objets sources.

30 * le procédé est mis en œuvre dans des moyens de traitement en cascade aptes à fournir les identifiants d'objets sources par parcours de proche en proche selon les transclusions.

35 * au moins à certaines transclusions sont associées des informations de transformation de l'objet, ces informations désignant chacune une fonction de transformation prédéterminée à appliquer à l'objet en tant qu'étape initiale de la transclusion.

* une transformation est apte à être appliquée sur des attributs d'objet tels que des attributs de contenu, de position, de méta-données et de transformations portant sur des transclusions internes aux objets.

40 * les ressources sont des feuilles de calcul et en ce que les objets sont des cellules ou groupes de cellules de la feuille.

* les formules associées aux cellules d'un groupe de cellules transclus sont considérées comme des transclusions internes au groupe.

45 * l'obtention d'une transclusion interne dans un groupe de cellules transclus comprend l'application au programme de feuille de calcul d'instructions de copier/coller à partir d'une transclusion interne donnée du groupe de cellules source.

Un douzième aspect de l'invention propose un procédé pour gérer un ensemble d'objets d'informations dans un environnement informatique, certains objets étant des objets conteneurs susceptibles de contenir d'autres objets, et l'environnement informatique comprenant une première structure d'objets et une deuxième structure d'objets transcluse à partir de la première, caractérisé en ce que :

- au moins certains objets de la première structure possèdent des propriétés ;

- au moins certains objets co : a deuxième structure possèdent des contraintes,

et en ce que le procédé comprend l'étape consistant, lors de l'opération de transclusion de la première structure vers la première structure, à présenter les objets de la première structure dans les objets conteneurs de la deuxième structure fonction de l'adéquation des propriétés des objets de la première structure avec les contraintes des objets conteneurs de la deuxième structure.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

* le procédé comprend en outre l'étape consistant, lors de la modification d'une propriété d'un objet de la première structure, à vérifier l'adéquation de la propriété modifiée avec les contraintes des objets conteneurs de la deuxième structure, et à déplacer le cas échéant l'objet correspondant présenté dans la deuxième structure.

* le procédé comprend en outre l'étape consistant, lors de la modification d'une contrainte d'un objet conteneur de la deuxième structure, à vérifier l'adéquation des propriétés des objets présentés dans cet objet conteneur avec la contrainte modifiée de celui-ci, et à déplacer le cas échéant le ou les objets considérés.

Selon un treizième aspect de l'invention, on prévoit un procédé pour modifier des contenus constitués par des flux de données balisés auxquels on accède selon un protocole de syndication, chaque contenu étant accessible sous forme d'un canal qui comprend plusieurs éléments d'information possédant chacun un identifiant unique, caractérisé en ce qu'il comprend les étapes suivantes :

- délimitation d'un élément d'information ou d'une partie d'un élément à modifier,
- application de la modification à l'élément d'information ou à la partie d'élément,
- élaboration d'un contenu contenant d'une part l'élément d'information ou partie d'élément modifié délimité par des balises spécifiques, et d'autre part l'élément d'information d'origine délimités par ses balises d'origine,
de sorte à éviter une perte d'information lorsque le flux est traité selon un protocole qui ne gère pas lesdites balises spécifiques.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

* à l'élément d'information ou partie d'élément modifié est associée une information indiquant s'il remplace l'élément d'information ou partie d'élément d'origine ou s'il constitue une alternative à l'élément d'information ou partie d'élément d'origine, et en ce qu'il comprend dans ce dernier cas une étape de signalisation de l'existence de l'alternative.

* le procédé comprend en outre une étape consistant, en réponse à la modification d'un élément d'information ou d'une partie d'élément, à signaler cette modification à un serveur d'où provient le flux de données.

* les ressources sont des flux de données balisés auxquels on accède selon un protocole de syndication, et en ce que les objets desdites ressources sont des éléments d'information desdites ressources ou des parties desdits éléments d'information (applicable à l'ensemble des procédés susmentionnés).

Selon un quatorzième aspect de l'invention est proposé un procédé pour transformer une première ressource comprenant une structure arborescente d'objets en une deuxième ressource comprenant une structure arborescente d'objets, sur la base de transformations pouvant être effectuées sur des objets situés à des niveaux différents de l'arborescence, caractérisé en ce que, lorsqu'il existe une première transformation sur un certain objet englobant au moins un objet descendant et une seconde transformation sur ledit objet descendant, la seconde transformation est exécutée prioritairement à la première transformation.

Selon un quinzième aspect, on prévoit un procédé pour l'accès par un utilisateur à des objets d'informations engendrés par un système collaboratif où différents groupes d'utilisateurs peuvent contribuer à former respectivement différentes versions d'objets à partir d'objets de départ communs, caractérisé en ce que l'utilisateur est susceptible d'appartenir simultanément à différents groupes, et en ce qu'il comprend les étapes consistant :

- en fonction d'un identifiant d'un des groupes sous lequel l'utilisateur réalise l'accès, présenter les versions correspondantes des objets partir des objets de départ,
- en fonction d'une information de mode d'accès contributif/non contributif par l'utilisateur, prendre en compte ou non dans le système collaboratif au niveau du groupe considéré les actions modificatives subséquentes de l'utilisateur sur les versions d'objets auxquelles il a accédé.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

- * en mode d'accès non contributif lors de l'accès sous un groupe donné, les actions modificatives peuvent être prises en compte par défaut dans d'autres groupes.

Selon un seizième aspect de l'invention, un procédé pour l'accès par des utilisateurs à des objets d'informations engendrés par un système collaboratif où différents utilisateurs peuvent contribuer à former respectivement différentes versions d'objets, et où les utilisateurs peuvent sélectivement accepter ou refuser des versions, est caractérisé en ce qu'il comprend des moyens de comptabilisation des contributions des utilisateurs et des acceptations/refus par d'autres utilisateurs des versions d'objets contenant ces contributions.

- en fonction d'une information de mode d'accès contributif/non contributif par l'utilisateur, prendre en compte ou non dans le système collaboratif au niveau du groupe considéré les actions modificatives subséquentes de l'utilisateur sur les versions d'objets auxquelles il a accédé.

Un dix-septième aspect de l'invention prévoit un procédé pour rendre accessible à un groupe d'utilisateurs au sein d'un environnement informatique une ressource contenant une pluralité d'objets susceptibles d'être modifiés par les utilisateurs pour obtenir différentes versions desdits objets, caractérisé en ce qu'il comprend :

- pour au moins certains objets, affecter aux différentes versions desdits objets des clés de chiffrement/déchiffrement différentes,
- prévoir des versions desdits objets chiffrées avec les clés respectives,
- lors de l'accès à la ressource par un utilisateur, effectuer le déchiffrement des versions d'objets chiffrées appartenant à la ressource à l'aide des clés de déchiffrement respectives,
- lors la modification d'un tel objet par un utilisateur, créer une nouvelle version de l'objet, à laquelle une nouvelle clé est affectée.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

- * le procédé comprend une étape de fourniture sélective des clés de déchiffrement des objets aux utilisateurs en fonction d'informations d'habilitation desdits utilisateurs.

* les habilitations sont traitées par groupes d'utilisateurs, deux utilisateurs d'un même groupe ayant les mêmes droits.

* les versions d'un objet sont constituées par des mises à jours successives dudit objet au niveau de postes clients de l'environnement informatique, et en ce que l'étape de création d'une nouvelle clé lors de la création d'une nouvelle version de l'objet est effectuée au niveau du poste client, une étape subséquent de transmission de la nouvelle clé à un serveur de clés étant prévue.

* les versions d'un objet sont constituées par des versions à contenus différents dudit objet, coexistant simultanément et réservées à des groupes différents d'utilisateurs, et en ce

qu'une communication d'un tel objet par d'un deuxième groupe comprend :

par d'un premier groupe à un utilisateur

- la fourniture à l'utilisateur du deuxième groupe, par tout canal approprié, d'un objet multi-versions contenant le contenu de l'objet réservé au deuxième groupe, à l'état chiffré, et

5 - la fourniture indépendante, par un serveur de clés, à l'utilisateur du deuxième groupe, d'une clé de déchiffrement permettant le déchiffrement dudit contenu de l'objet réservé au deuxième groupe.

10 * le procédé comprend une étape consistant à découper l'objet multi-versions en parties homogènes du point de vue des droits d'accès, chaque partie contenant des informations de positionnement permettant son remplacement dans la structure, de telle sorte que des manipulations puissent être effectuées sur l'objet multi-versions même en l'absence de droit d'accès à la totalité de ses versions.

15 Enfin selon un dix-huitième aspect de l'invention, on propose un procédé pour rendre accessible à différents groupes d'utilisateurs (ERC) au sein d'un environnement informatique une ressource contenant une pluralité d'objets susceptibles d'être modifiés par les utilisateurs pour obtenir différentes versions desdits objets, caractérisé en ce que chaque objet est susceptible d'exister dans différentes versions correspondant à des groupes d'utilisateurs différents et descendantes d'un objet multi-versions les englobant, et en ce qu'il comprend, 20 lors de l'accès par un utilisateur à un objet, la présentation à celui-ci des versions de l'objet correspondant aux groupes auxquels il appartient, chaque version d'objet étant obtenue par transformation à partir d'une autre version dudit objet.

Certains aspects préférés mais non limitatifs de ce procédé sont les suivants :

25 * le procédé est mis en œuvre à l'aide d'un procédé de gestion selon la revendication 62, des transformations prédéterminées spécifiques permettant d'obtenir chacune des versions de l'objet.

* une action de transclusion par un utilisateur d'un objet appartenant à un objet multi-versions provoque une transclusion de l'ensemble de l'objet multi-versions.

30 * chaque objet est également susceptible d'exister au sein d'un même groupe d'utilisateurs dans différentes versions correspondant à différents utilisateurs, ces versions pouvant être également présentées lors de l'accès à l'objet par les utilisateurs dudit groupe.

35 * le procédé comprend en outre la mise en œuvre de moyens de calcul de scores de pertinences des différentes versions correspondant aux différents groupes et/ou aux différents utilisateurs pour présenter sélectivement les versions les plus pertinentes.

Définitions préliminaires

Objet, Ressource

40 Toute information accessible et identifiée de manière unique est un *objet*. Par exemple un élément XML identifié de manière unique est un objet. Un paragraphe contenu dans un document en format Word n'est pas un objet dans la mesure où il n'a pas d'identifiant utilisable. On va mettre en œuvre des moyens de le « réifier » (réifier signifie en faire un objet). Un objet réifié est typiquement sélectionnable par clic¹, déplaçable ou copiable (ou encore « transluable » – la transclusion est définie plus loin), notamment par glisser-déposer.

¹ (en cliquant sur l'objet lui-même, ou encore sur une poignée graphique qui apparaît au voisinage de l'objet quand la ressource contenant l'objet est visualisée dans un mode spécial, ou encore en sélectionnant l'objet et en actionnant un bouton, etc.)

Une *ressource* est un objet géré par : niveau système tel que le système d'exploitation ou le serveur Internet. Les ressources typiques sont les pages Web, les fichiers et les répertoires.²

- 5 A noter que le *propriétaire* d'une ressource n'est pas forcément la personne qui détient des droits de propriété intellectuelle sur son contenu, mais plutôt le créateur de cette ressource (cette dernière peut par exemple présenter le même contenu qu'une ressource appartenant à quelqu'un d'autre, ce contenu ayant été obtenu par transclusion ou par copie).

- 10 Par soucis de concision on utilisera parfois le terme « ressource » à la place de « propriétaire de ressource ». Par exemple, on dira « R3 accepte b4 » pour exprimer que « le propriétaire de la ressource R3 accepte la modification faite sur l'objet b et qui a résulté en l'objet b4 ».

ERC, Serveur collaboratif

- 15 On définit un *Ensemble de Ressources Collaboratives* (ERC) comme un ensemble de ressources identifiables par un type de serveur particulier (décrit plus loin) que l'on appelle *serveur collaboratif*³, ou par un dispositif analogue.⁴ Par ailleurs, un ERC est caractérisé en ce que toutes les ressources dudit ensemble sont accessibles par un ensemble défini d'utilisateurs. Dans la suite, le terme ERC désigne tout autant l'ERC que le groupe d'utilisateurs qui y a accès. On dit alors qu'un utilisateur « appartient » à un ERC.

A noter qu'un ERC est géré par au plus⁵ un serveur collaboratif (ou dispositif analogue), mais un serveur collaboratif peut gérer plusieurs ERC.

- 20 On dit qu'une ressource *appartient* à (ou fait partie de) un ERC si elle est identifiée comme telle par le serveur collaboratif (ou dispositif analogue) qui gère cet ensemble⁶.

- 25 On dit que l'accès à une ressource (ou ressource collaborative) se fait à travers (ou dans le cadre de) un ERC donné⁷ (afin de bénéficier des fonctionnalités inhérentes aux ERC qui sont décrites ci-après).⁸ L'utilisateur peut aussi accéder à une ressource en laissant le système déterminer les serveurs collaboratifs susceptibles d'identifier cette ressource.

On dira indifféremment « serveur collaboratif » ou « serveur de collaboration ».

Utilisateur

Par « utilisateur » on entend « utilisateur ou programme » ; notamment, par « interface utilisateur » on entend « interface utilisateur ou programmation » (sauf contre sens). Le terme

² Le « niveau système » est subjectif et relatif au point de vue dans lequel on se place. A noter aussi que les ressources contiennent des objets qui en général ne sont pas des ressources eux-mêmes, mais il y a beaucoup d'exemples où ceci n'est pas le cas, notamment les répertoires (qui contiennent eux-mêmes des ressources) et les courriers électroniques (qui sont des ressources et les pièces jointes qu'ils « contiennent » le soient aussi).

³ Par abus de langage, on dira qu'une ressource est « identifiée par un ERC donné » plutôt que « identifié (c'est-à-dire reconnu) comme appartenant à un ERC donné ».

⁴ Voir la section présentant l'architecture en cascade. Par ailleurs, on montre à la section « Extensions aux protocoles de syndication actuels » (vers la fin du document) qu'un ERC peut aussi être un ensemble de ressources identifiables dans un réseau de syndication (ou partage), même sans l'existence d'un *Serveur Collaboratif* proprement dit. Toutefois on décrit aussi que la disponibilité d'au moins un serveur collaboratif offre des avantages.

⁵ Toutefois, la répartition de la gestion d'un ERC sur plusieurs serveurs est également possible.

⁶ Et une ressource peut appartenir à plusieurs ERC.

⁷ Dans le cas où la ressource n'est pas identifiée par l'ERC donné, ce dernier est juste ignoré. L'utilisateur peut se placer dans plusieurs ERC à la fois. L'utilisateur peut aussi basculer d'un ERC à l'autre en utilisant un moyen interactif prévu à cet effet. Les actions du système sont effectuées séparément pour chaque ERC et les résultats peuvent être présentés séparément ou en combinaison.

⁸ A noter aussi qu'un serveur collaboratif ne stocke (pas forcément) les objets qu'il gère mais (plutôt) ses identifiants.

« utilisateur » peut aussi signifier « (exte
le poste client) ».

util de manipulation d'information (sur

Par « l'utilisateur peut sélectionner... » on entend que l'utilisateur peut choisir une option soit au moment même soit en configurant/paramétrant le système à l'avance (sauf contre sens).

- 5 Par « l'utilisateur ayant les droits d'accès.. », on entend l'utilisateur individuel ou un groupe d'utilisateurs, ou encore un rôle d'utilisateur, ayant les droits d'accès en question (sauf contre sens).

- 10 Enfin, on va donner des exemples où chaque utilisateur ayant contribué une modification à une ressource possède une version propre de cette ressource, ses modifications étant locales à cette version. Ces exemples sont donnés pour faciliter la compréhension et l'on entend bien qu'une même version d'une ressource peut en général être modifiée par plusieurs utilisateurs et qu'un utilisateur peut en modifier plusieurs.

Description

Introduction

- 15 Un serveur collaboratif⁹ permet à un ensemble d'utilisateurs de manipuler des ressources en commun similairement à un « wiki » (procédé connu en soi ; voir notamment sur <http://fr.wikipedia.org/wiki/Wiki>).

- 20 La différence avec les wikis est que d'une part le procédé de l'invention permet de modifier ainsi toute ressource présente dans l'Internet (qui n'est pas nécessairement prévue pour) et d'autre part qu'il peut gérer simultanément des variantes pour des parties de ces ressources (exprimant par exemple plusieurs points de vue).

Dans le cas de la modification d'une ressource externe¹⁰, les modifications ayant eu lieu dans cette dernière sont répercutées dans ses différentes versions vues en utilisant le procédé.

- 25 Le procédé s'utilise via les outils de navigation usuels, munis d'extensions effectuant l'interface avec un serveur collaboratif.

Il est similaire aux wikis dans le sens où dès qu'un utilisateur modifie un élément d'une ressource cette modification est immédiatement montrée aux autres utilisateurs de l'ERC lorsque ces derniers vont consulter cette ressource (s'ils se placent dans le même point de vue).

- 30 On va avoir un (ou plusieurs) ensemble(s) de ressources collaboratives public(s), et en parallèle des plus petits groupes peuvent se créer leurs ERC privés¹¹.

Illustration du principe du procédé

Les figures 1-a à 1-g illustrent différentes étapes d'utilisation du procédé et permettent de le comprendre intuitivement, avant même de lire sa description qui va suivre.

- 35 Par convention, la lettre v suivie d'un chiffre (v1) représente une version d'une ressource collaborative. Une voyelle suivie d'un chiffre (a1) représente une version d'un objet (a) ; elle est soulignée (a1) quand elle représente la version la plus récente¹².

⁹ On considère ici une conception mettant en œuvre un serveur collaboratif ; on présentera vers la fin de ce document une autre approche (du même procédé) qui ne nécessite pas un tel serveur (voir section « Extensions aux protocoles de syndication actuels », voir aussi la section présentant l'« Architecture en Cascade »).

¹⁰ (par exemple une page Web publique qui n'a pas été créée en utilisant le procédé de l'invention)

¹¹ Un cas extrême est la création d'un ERC par une seule personne pour son usage personnel, si elle souhaite utiliser les fonctions de modification des objets sans pour autant vouloir partager ces modifications

¹² (la plus élaborée)

La figure 1-a présente une première vers ressource et une première version (a1) d'un objet que cette ressource possède.

On suppose ici que l'utilisateur veut modifier cet objet ; une nouvelle version (v2) est alors dérivée de v1 ; c'est ce que montre la figure 1-b. Par convention, les figures ne présentent que les objets créés ou modifiés.

La figure 1-c montre la situation après que l'objet a ait été *modifié* dans v2. Lorsque l'utilisateur visionne v1 ou v2, il voit automatiquement la version la plus récente de cet objet, c'est-à-dire celle de v2, qui est donc soulignée dans la figure (a2).

La figure 1-d présente le cas où le propriétaire de v1 *refuse* la version la plus récente de l'objet a (c'est-à-dire la version a2, provenant de v2). En conséquence, une nouvelle « branche » est créée pour cet objet et a2 y est mise. Quant à a1, la version de cet objet dans v1, elle se retrouve alors seule dans la première branche et redevient donc la plus récente (pour cette branche).

Les branches d'un objet, autres que la branche couramment activée, sont présentées à l'utilisateur comme versions alternatives de cet objet.¹³

La figure 1-e présente le cas où l'utilisateur visualisant v1 sélectionne, pour l'objet a, la deuxième branche (dont la version la plus récente a2 lui est présentée comme version alternative, c'est-à-dire en tant qu'option sélectionnable). On dit que l'utilisateur « *active* » la deuxième branche (il fait cela simplement en sélectionnant la version alternative en question). Dans la figure, on représente ceci par un déplacement du rectangle vers la branche de droite alors que l'objet créé initialement reste bien dans la branche de gauche.

La figure 1-f présente le cas où l'utilisateur *modifie* dans v1 l'objet a se trouvant dans la deuxième branche. Il crée ainsi la version a3 qui devient alors le plus récent dans la deuxième branche. En conséquence, quand un utilisateur visionne v2 en ayant activé la deuxième branche, il voit la version la plus récente de cette dernière, donc celle de v1 (c'est-à-dire a3).

La figure 1-g présente le cas où l'utilisateur visualisant v2 *active* la première branche. Il voit donc la version (a1) de v1 qui est la plus récente pour la première branche de l'objet a.

On va maintenant reprendre la description en détail.

Versions d'une ressource collaborative, branches

Pour faciliter la compréhension, dans les exemples et les explications qui suivent, chaque utilisateur ayant contribué une modification à une ressource collaborative possède une version propre de cette ressource collaborative. On dira que ces modifications sont locales à cette version de la ressource collaborative.

En l'absence des "branches" qui seront décrites ci-après, lorsque l'utilisateur visionne une ressource collaborative il verra pour chaque objet la *version la plus récente* de cet objet. Ceci signifie que toutes les versions de la ressource collaborative sont présentées de façon identique¹⁴.

L'utilisateur a cependant la possibilité, lorsqu'il visionne une version d'une ressource collaborative, de demander à voir, lorsqu'elles existent, les modifications locales à cette version.

¹³ Ce sont les versions d'objet les plus récentes de ces autres branches qui lui sont présentées.

¹⁴ On peut cependant envisager que le système offre aux utilisateurs d'avoir des objets ou modifications "secrets" dans leurs versions de la ressource collaborative

Dans ce qui suit, on va d'abord uniquement les modifications non-structurelles, i.e. qui se contentent de modifier les valeurs ("innertext") des objets sans en créer, déplacer ou supprimer.

5 Dans les figures 2-a à 6-b, on utilise des tableaux où les lignes sont des différentes versions de la ressource collaborative (comme dans les figures 1-a à 1-g) et les colonnes sont des différents objets contenus dans la ressource collaborative. Les valeurs inscrites à l'intérieur du tableau donnent les versions locales de ces objets - une case vide indique une absence de version locale. Chaque colonne contient une valeur soulignée qui désigne la version la plus récente dans cette colonne.

10 Dans les figures 2-a à 2-d, on considère une ressource collaborative qui contient trois objets a, b et c. Il en existe tout d'abord une seule version v1, où les valeurs des objets sont respectivement a1, b1 et c1 (figure 2-a).

15 Ensuite (figure 2-b) un utilisateur modifie l'objet a et y écrit a2 à la place. Ceci provoque la création de la version v2 de la ressource collaborative, le premier contenu a1 restant inchangé dans la version v1. Cependant qu'un utilisateur visionne v1 ou v2 il verra a2 comme contenu du premier objet car ceci est la valeur la plus récente.

Ensuite (figure 2-c) un autre utilisateur modifie les objets a et b, et y écrit a3 et b3, respectivement. Ceci crée une troisième version de la ressource collaborative, v3.

20 Ensuite (figure 2-d) le propriétaire de v2 modifie l'objet a et y inscrit la valeur a4. Étant donné qu'il travaille dans sa propre version, il remplace l'ancienne valeur locale a2. Si un utilisateur visionne une quelconque version de la ressource collaborative il verra les valeurs a4, b3 et c1, respectivement, pour les trois objets.

25 Le procédé offre la possibilité de refuser des versions d'objet (i.e. des modifications d'objet effectuées par un autre utilisateur). La conséquence de ce refus est de créer une branche pour chaque objet dont la version de l'autre utilisateur a été refusée, et de déplacer ces versions refusées dans les nouvelles branches. Parmi les autres versions disponibles dans la même branche, ce sera la plus récente qui sera affichée après le refus.

30 Chaque objet d'une ressource collaborative peut ainsi être partitionné en plusieurs branches, et chaque utilisateur a ensuite la possibilité de choisir (pour chaque objet) quelle branche il préfère (ceci est présenté sous forme de "versions alternatives", comme décrit à la section suivante). Ainsi, chaque version de la ressource collaborative spécifie pour chaque objet quelle (il y en a exactement une) branche y est activée.

35 Une modification ayant lieu dans une branche d'un objet ne sera pas répercutée dans les versions où cette branche n'est pas activée (sauf si l'utilisateur demande de rendre visible les autres branches, cf. la section suivante).

Les différentes branches d'un objet sont traitées indépendamment en ce qui concerne les modifications, comme s'il s'agissait d'objets différents.

En résumé :

- 40
- chaque ressource collaborative contient un certain nombre d'objets
 - chaque objet possède un certain nombre de branches
 - chaque ressource collaborative possède un certain nombre de versions

¹⁵ (au sens d'XML)

- chaque version d'une ressource collaborative pour chaque objet quelle branche en est active.

Lorsqu'un utilisateur visionne une version de la ressource collaborative, pour chaque objet sa version la plus récente dans la branche active est présentée.

- 5 Optionnellement, pour chaque branche de chaque objet, chaque version de ressource collaborative peut en fournir une version locale.

- 10 L'exemple des figures 3-a à 3-d continuent le cas présenté à la figure 2-d. On représente les différentes branches d'un objet par autant de colonnes adjacentes. Une case encadrée (rectangle) indique que la branche correspondant à la colonne est active dans la version de la ressource collaborative correspondant à la ligne.

- 15 Figure 3-a : Le propriétaire de v2 refuse la modification b3 (de v3). Ceci crée une nouvelle branche (branche 2) dans laquelle est envoyé b3, et b1 redevient la version la plus récente dans l'ancienne branche. Seul v2 active la nouvelle branche. À ce moment, les versions v1 et v2 affichent a4, b1 et c1 comme valeurs pour les trois objets et v3 affiche a4, b3 et c1.

- Figure 3-b : Le propriétaire de v3 se venge et refuse la valeur a4 de l'objet a, qui est à son tour envoyée dans une nouvelle branche. La version la plus récente dans la première branche de a redevient a3. Les valeurs affichées pour les trois versions sont donc respectivement: a3 b1 c1 , a4 b1 c1, a3 b3 c1.

- 20 Figure 3-c : v3 modifie sa version de b, dans sa branche, et y écrit b4. Cependant il est le seul à le voir car il est le seul à avoir cette branche active.

- Figure 3-d : v3 sélectionne la seconde branche de a. Ceci a pour effet que visionner v3 montre a4, b4 et c1 comme valeurs. Puis il modifie a pour y écrire a5. A ce moment v3 possède deux versions locale de a (une pour chaque branche), dont la deuxième est visible. Les valeurs affichées pour les trois versions sont maintenant a3 b1 c1 , a5 b1 c1, a5 b4 c1.
- 25

Visionner les versions alternatives d'une ressource collaborative

Le procédé permet à l'utilisateur de visionner (et activer) les différentes branches¹⁶ existant pour chaque objet. Lorsque qu'une branche d'un objet est montrée, c'est la version la plus récente se trouvant dans cette branche qui est affichée par défaut.

- 30 Il offre à l'utilisateur différentes méthodes pour ce faire.

Premièrement l'utilisateur peut faire afficher, en relation¹⁷ avec chaque objet, la liste des versions alternatives de cet objet¹⁸.

- 35 Deuxièmement l'utilisateur peut demander à voir les différentes versions de la ressource collaborative (qui diffèrent les unes des autres par les branches actives) - ceci permet à l'utilisateur de voir les corrélations entre les différentes branches des différents objets de la ressource collaborative.

- 40 L'utilisateur peut également utiliser une méthode mixte : lorsqu'il existe un grand nombre de versions de la ressource collaborative l'utilisateur peut "filter" parmi elles en choisissant tout d'abord une branche pour un objet particulier (ou plusieurs), puis en visionnant les différentes versions¹⁹ de la ressource collaborative ayant activé la même branche pour cet objet. Le système va de plus éviter de distinguer deux versions ayant la même combinaison de branches

¹⁶ (c'est-à-dire les différentes versions alternatives)

¹⁷ (par exemple à côté de chaque objet)

¹⁸ (par exemple directement dans la ressource collaborative ou via une fenêtre en pop-up)

¹⁹ (ou plutôt en visionnant les configurations de branches pour ces différentes versions)

à l'intérieur de la ressource considérée (et encore plus l'utilisateur peut même demander de n'avoir les variantes que d'une partie de la ressource).

Enfin, dans une version de ressource collaborative, l'utilisateur peut entériner la version d'objet la plus récente (telle qu'elle lui a été présentée) d'une branche, de manière à en faire sa version d'objet locale pour cette branche (et remplacer celle qui y figurait le cas échéant). Alternativement il peut refuser la version la plus récente de l'objet en question (ainsi que les autres versions moins récentes qui lui sont présentées en suivant), jusqu'à en entériner finalement une (qui peut être sa propre version ou une autre). Dans la mesure où il a ainsi entériné plusieurs objets, il communique (notamment au moment de l'enregistrement de la version de ressource collaborative en question) un jeu de versions locales d'objets qui lui paraît cohérent.

Lors de l'utilisation de la méthode mixte décrite ci-dessus, le filtrage peut alors se faire en considérant des jeux de versions d'objets entérinées. On exploite ainsi les contributions des utilisateurs mêmes, pour assurer la cohérence des branches activées entre elles.

15 Dérivation

On offre à l'utilisateur plusieurs moyens d'éviter le travail de sélectionner objet par objet les branches qu'il préfère.

D'une part en visionnant une version différente de la ressource collaborative, il peut demander de copier dans sa version ces différents choix de branches.

Un autre moyen est de permettre à une version de la ressource collaborative (la ressource collaborative aval) de « dériver » d'une autre (amont) de manière à ce que lorsque la version amont change de branche pour l'un de ses objets, la version aval va automatiquement adopter cette nouvelle branche, sauf si son propriétaire a explicitement déclaré quelle branche il préférerait pour cet objet.

Par défaut, chaque version de ressource collaborative dérive de la version dont la modification lui a donnée naissance.

En plus de ceci, un autre procédé consiste à chercher pour un utilisateur quelle version est proche (dans ses choix de branches - ceci peut se mesurer en regardant la proportion de choix de branches qui correspondent par rapport au nombre total d'objets) de celle de l'utilisateur, et proposer à ce dernier de placer sa version en dérivé de cette autre version.

Une telle « dérivation à posteriori » peut aussi se faire par d'autres méthodes de recherche d'affinité, ou même manuellement par l'utilisateur (notamment dans le cas d'un élève qui dérive sa version de celle de son maître ou d'un directeur qui dérive de sa secrétaire).

L'exemple des figures 4-a à 4-d reprend exactement les mêmes opérations que l'exemple des figures 3-a à 3-d mais suppose que les relations de dérivation ont été placées comme suit: v3 dérive de v2 puis v1 dérive de v3. Ceci implique que les changements de branche ayant lieu dans v2 seront répercutés dans v3 et les changements ayant lieu dans cette dernière seront répercutés dans v1, à l'exception que des sélections de branche explicites ne seront pas remplacées ainsi.

Figure 4-a : Le propriétaire de v2 refuse la modification b3 (de v3). Ceci crée une nouvelle branche dans laquelle est envoyée b3. v2 va donc explicitement choisir la première branche et v3 va explicitement choisir la seconde. Par contre v1 n'effectue pas de choix explicite et donc suit celui de v3 (i.e. la seconde branche). À ce moment, seule la version v2 affiche a4, b1 et c1 comme valeurs pour les trois objets, alors que v1 et v3 affichent a4, b3 et c1.

Figure 4-b : Le propriétaire de v3 refuse la valeur a4 de l'objet a, qui est à son tour envoyée dans une nouvelle branche. v3 va donc explicitement choisir la première branche et v2 va

explicitement choisir la seconde. v1 e: 17 r3, et donc ne suit pas v2 dans son changement de branche. Les valeurs affichées pour les trois versions sont donc respectivement: a3 b1 c1, a4 b1 c1, a3 b3 c1.

- 5 Figure 4-c : v3 modifie sa version de b, dans sa branche, et y écrit b4. Contrairement à l'exemple précédent, il n'est pas le seul à le voir car v1 en dérive et a également (implicitement) activé cette branche (v1 et v3 verront donc cette nouvelle valeur alors que v2 continuera à voir l'ancienne valeur b1).

- 10 Figure 4-d : v3 sélectionne la seconde branche de a. De même qu'à l'étape 4-a, v1 va implicitement activer cette seconde branche. Les trois versions ont donc activé la seconde branche de a et y voient la valeur a4. Puis v3 modifie l'objet a pour y écrire a5. Les valeurs affichées pour les trois versions sont maintenant a5 b4 c1, a5 b1 c1, a5 b4 c1.

Catégorisation des versions de ressources collaboratives

On décrit ici un perfectionnement permettant de qualifier les modifications effectuées par les différents utilisateurs à l'aide de mots clés ou de catégories.

- 15 Comme déjà décrit, les différentes branches d'un objet peuvent représenter plusieurs points de vue. On permet donc aux utilisateurs d'attacher aux branches d'objets des « catégories » décrivant la nature de ce point de vue. Chaque utilisateur ayant la possibilité de changer le contenu d'un objet a également la possibilité de changer à quelle catégorie appartient sa version de l'objet.
- 20 Afin d'assurer que tous les utilisateurs utilisent un vocabulaire commun, le système tient à jour une hiérarchie de catégories (taxonomie), de l'intérieur de laquelle les utilisateurs peuvent sélectionner les qualificatifs pour leurs modifications. Cette hiérarchie est globale à l'ERC, mais on peut autoriser aux utilisateurs de la modifier, notamment en ajoutant ou identifiant des catégories²⁰.
- 25 Une fois que les utilisateurs ont ainsi étiqueté leurs modifications, on peut exploiter cette information et permettre aux autres utilisateurs de filtrer les versions disponibles en fonction des catégories. Ceci fournit donc une quatrième méthode pour naviguer dans les différentes versions d'une ressource collaborative, en plus de celles proposées à la section « Sélectionner des branches automatiquement » ci-dessus. On va maintenant montrer comment cela fonctionne.
- 30

Étant donné que la catégorisation d'une modification est une option et non une obligation le système procède par élimination des branches ayant été catégorisées autrement :

1. Un utilisateur va consulter une ressource et déclare s'intéresser à une ou plusieurs catégories en particulier.
- 35 2. Le système élimine (de l'ensemble des choix possibles) les versions ayant choisi au moins une branche dont le contenu le plus récent est catégorisé de façon contradictoire aux catégories demandées.
3. Puis l'utilisateur peut explorer les versions qui restent valable selon la dernière méthode décrite à la section « Sélectionner des branches automatiquement ».

40 Modifications structurelles, créneaux

Nous allons maintenant décrire comment généraliser ce mécanisme à tout type de modification, incluant création, déplacement et suppression d'objets.

²⁰ (ou au sens des folksonomies)

On prend ici la convention que la relation d'un objet à un autre de deux objets n'est pas stockée dans l'enfant mais dans le parent. Cette information est représentée sous forme de créneaux²¹ se trouvant dans l'"innertext" indiquant quel enfant doit s'y trouver.

5 Ainsi le déplacement d'un objet d'un point à un autre sans changer de parent consiste à changer le parent uniquement ; un déplacement d'objet ayant pour effet de changer quel est le parent de ce dernier (sans pour autant changer de ressource collaborative) consiste en la modification de chacun des parents. A noter que dans ces deux cas l'objet déplacé n'est pas modifié, seuls ses parents source et destination le sont.

10 La suppression d'un objet consiste simplement à en enlever la référence dans le parent. Ici encore on considère que seul le parent est modifié, pas l'objet lui-même. Ces considérations sont importantes quand on considère que par exemple un objet peut être supprimé dans une branche (du parent) mais pas dans une autre²².

15 La création d'objet, se fait (du point de vue du système) en deux étapes : tout d'abord un nouvel objet est créé au niveau de la ressource collaborative (car un objet est un concept global à la ressource collaborative, et non spécifique à une version de ressource collaborative). Puis l'objet où l'utilisateur a demandé à créer un enfant est modifié pour contenir une référence à ce nouvel objet. Ainsi si par exemple une version souhaite refuser cette création elle va seulement refuser la modification du parent, ce qui aura pour effet de rendre le nouvel objet invisible.

20 L'exemple des figures 5-a à 5-d continue le cas de la figure 4-d, en faisant l'hypothèse supplémentaire que b et c sont des enfants de a, c'est-à-dire que toutes les versions de a contiennent deux créneaux, un pour b et un pour c. Par exemple on pourrait dire que a3 a pour valeur " habite à <c>", que b4 a pour valeur "Arthur" et c1 a pour valeur "Amsterdam", et ainsi de suite.

25 Figure 5-a : Le propriétaire de v1 demande au système de créer un nouvel enfant pour b. Pour ce faire il place un créneau dans sa branche de b (il a sélectionné la seconde branche, donc va ajouter un créneau à b4. La version de b4 étendue d'un nouveau créneau est donc locale à v1. On l'appelle b5). Le système va créer un nouvel objet vide qu'on appelle d, et cette valeur initiale vide est appelée d1. L'objet d1 est créé et activé dans toutes les versions, mais comme
30 il est seulement référencé depuis b5, seuls v1 et v3 verront l'objet d.

Figure 5-b : Le propriétaire de v3 déplace maintenant l'objet d comme enfant de c1. Ceci signifie qu'il supprime la référence se trouvant dans b5 et ajoute une référence dans c1. Il crée ainsi de nouvelles versions de ces deux objets, b6 et c6. Une conséquence de ceci est que maintenant d est rendu visible à v2 également, car il a été introduit dans une branche d'objet
35 qui était active dans sa version. A noter également que d n'a pas été modifié dans l'opération, il n'en existe toujours qu'une version d1, appartenant à v1.

Figure 5-c : Le propriétaire de v2 refuse l'irruption de cet objet. Pour ce faire il ne refuse pas d1 lui-même (car un refus signifie en fait le refus du contenu de l'objet, et non l'objet lui-même), mais la nouvelle version c6 de c. Ceci a pour effet de créer une nouvelle branche de c, l'ancienne branche reprenant l'ancienne valeur c1 dépourvue de référence à d. A noter que
40 comme v1 dérive de v3, il va suivre v3 dans la nouvelle branche, et le nouvel objet ne sera refusé que dans v2.

Figure 5-d : Le propriétaire de v1 supprime l'objet b. Pour ce faire il modifie l'objet a pour enlever le créneau pointant vers b. Ceci va créer une nouvelle version a6 de l'objet a. Comme

²¹ (« slot » en terminologie anglo-saxonne)

²² Afin d'éviter d'avoir ainsi des objets devenant inaccessibles (toutes leurs références ayant été supprimées) mais occupant de la mémoire, un "ramasse miettes" (garbage collector) qui détruira les objets l'étant plus référencés nulle part pourrait être nécessaire.

toutes les versions ont activé la seconde l eci a pour effet de faire disparaître b de toutes les versions. Cependant b n'est pas réellement supprimé car il est toujours référencé depuis a1, a3, a4 et a5. Donc si par exemple une quelconque des trois versions active la première branche de a alors a3 sera rendu visible, avec sa référence vers b. D'autre part si soit v2 soit v3 demande à voir sa version locale de la ressource collaborative alors encore une fois b sera rendu visible. Si par contre chacune de ces cinq versions de l'objet a supprime la référence vers l'objet b alors plus personne n'aura la possibilité d'atteindre ce dernier, et il pourra donc être supprimé de façon transparente (par le ramasse-miette).

A noter que dans certain cas, plusieurs références vers un même objet sont rendues visible simultanément. Par exemple si juste après l'étape la figure 5-a un utilisateur déplace b comme enfant de c alors un utilisateur ayant activé la première branche de a verra b à la fois comme enfant de a et de c.

Gestion des incohérences

Lorsqu'un utilisateur effectue un déplacement, il peut arriver que dans d'autres versions de la ressource collaborative (en fonction des branches choisies) cette opération soit vue comme une suppression, une création d'objet, et dans certains cas que le même objet apparaisse à plusieurs endroits.

Il existe des applications dans lesquelles ce type de phénomènes n'est pas problématique, mais l'utilisateur peut souhaiter placer un certain nombre de contraintes, par exemple qu'un certain objet doit apparaître au plus une fois, ou alors qu'une opération de déplacement doit, pour un autre utilisateur donné, soit n'avoir aucun effet visible, soit être vu comme un déplacement.

Afin de garantir ce genre de contraintes, tout en continuant à autoriser tout changement sur les objets, au cas où cette opération serait vue par un autre utilisateur d'une façon non autorisée, il s'agit de créer des nouvelles branches pour l'origine et/ou la destination du déplacement, dont la valeur contient celle avant le déplacement, et d'ensuite activer automatiquement cette ou ces nouvelles branches chez les utilisateurs en violation des contraintes.

À noter que ces contraintes, en particulier celles d'unicité de position d'un objet, vont interdire certaines combinaisons de branches. Si un utilisateur tente d'activer une combinaison non valable, le système adaptera automatiquement les autres branches afin d'arriver à une combinaison autorisée.

Dans l'exemple qui suit (voir les figures 6-a et 6-b) on va supposer qu'on a fourni la contrainte qu'un objet quelconque ne doit pas être visible à deux endroits simultanément.

Figure 6-a : Soient deux objets a et b, qui possèdent chacun deux branches (a1, a2 et b1, b2). On dit qu'un objet x (dont le contenu et les éventuelles branches ne nous intéressent pas pour cet exemple) est présent dans chaque branche de a. A ce moment la contrainte est satisfaite quel que soient les choix de branches car dans toutes les combinaisons x est enfant de a et seulement de a.

Figure 6-b : Supposons maintenant qu'un utilisateur déplace x de a1 vers b1 (dans la version v1). (Faisant ainsi les valeurs a1' et b1'). À ce moment il existe trois combinaisons autorisées, a1'-b1', a1'-b2, et a2-b2. Dans la seconde de ces combinaisons, x n'est pas visible du tout mais ce n'est pas en violation de la contrainte (mais on peut refaire le même exemple où la contrainte serait que x doit être visible exactement une fois et donc cette seconde combinaison serait alors interdite). S'il existe au moins une version de la ressource collaborative qui avait activé a2-b1 comme combinaison (v2), il faut faire en sorte que le déplacement n'ait pas d'effet visibles dans cette version. Ceci est fait en créant une nouvelle branche qui contient l'état original de b1 avant que ce dernier n'ait reçu l'objet x. Le système va alors, juste avant d'effectuer le déplacement, activer dans toutes les versions concernées (v2) la nouvelle

- 20
- branche b1. Si l'utilisateur demande à réactiver la branche b1' dans une de ces versions (v2) le système va le forcer à également activer la branche a1' de l'objet a afin d'éviter la combinaison a2-b1', et, symétriquement, si un utilisateur demande à activer la branche a2 dans une version (v1) où la branche b1' est active, le système forcera l'utilisateur à soit activer b1 soit b2 pour l'objet b.
- 5

Fusion de branches

On a décrit précédemment un mécanisme (le refus) permettant de créer de nouvelles branches pour un objet. Tel que décrite cette opération était irréversible car il n'y a pas moyen de supprimer ou fusionner des branches. C'est ce que cette section propose.

- 10 Un utilisateur peut exprimer son opinion que deux branches d'un objet doivent être fusionnées, notamment en copiant le contenu le plus récent de l'une des branches comme contenu pour une autre branche. En conséquence immédiate, qu'un utilisateur active l'une ou l'autre de ces deux branches après l'opération, il verra le même contenu.

- 15 Lorsque ceci se produit le système va fusionner les deux branches de cet objet. Il va donc devoir déterminer pour chaque version quel contenu placer dans la branche issue de la fusion (ou s'il faut ne pas mettre de contenu du tout).

Lorsqu'une version de la ressource collaborative possède une version locale dans au plus une des deux branches concernées, la fusion est triviale.

- 20 Par contre si une version possède un contenu local pour chacune des deux branches, le système doit choisir un des deux contenus. Par exemple dans le cas où l'une des deux branches est active, son contenu sera préféré, sinon on prend celui le plus récent.

- Afin d'éviter de détruire l'autre on peut créer pour cet objet une branche "inactive" (c'est-à-dire qui n'est jamais présentée à l'utilisateur en tant que telle mais qui sert à stocker les contenus ainsi "désactivés"). Lorsque le propriétaire de cette version visionne la ressource collaborative le système l'informe de la situation et lui propose différentes actions :
- 25

L'utilisateur peut détruire l'un ou l'autre de ces contenus

Il peut demander de recréer une branche pour y remettre son contenu (et les autres contenus "désactivés")

Cas de la ressource collaborative unique

- 30 On peut envisager une simplification du procédé où il existe en fait pour chaque « ERC » une seule "ressource collaborative" qui englobe tous les objets connus du système.

- Ensuite, contrairement à la méthode décrite ailleurs dans ce document, lorsque l'utilisateur va consulter ou modifier une ressource il le fait toujours à travers cette même ressource collaborative et doit dire au système quel objet particulier il souhaite traiter (avec ses descendants).
- 35

Ceci implique aussi (cf. section « Mise en œuvre » ci-dessous) qu'une ressource n'indique plus quelle est sa "racine", i.e. elle n'est plus liée à un objet particulier.

Ceci est intéressant si des ressources peuvent contenir d'autres ressources, par exemple un répertoire contenant des documents ou d'autres répertoires.

- 40 Il peut par exemple modifier l'organisation des fichiers et des répertoires, et ses modifications auront un effet cohérent quel que soit le répertoire que visionne un autre utilisateur. Si des ressources collaboratives différentes étaient utilisées pour chaque répertoire ce ne serait pas le cas - les changements ne seraient visibles qu'à condition de consulter la structure depuis le même point que lors de la modification.

Transclusions

On décrira le mécanisme de "transclusion" plus loin dans ce document, qui permet de construire des objets qui soient des versions modifiées d'autres objets.

- 5 Brièvement, une transclusion est similaire à une version de ressource à la différence que des objets vu à travers une transclusion peuvent cohabiter avec des objets non vus à travers une transclusion, ou vus à travers une autre transclusion etc.

Il est possible de regarder n'importe quel objet à travers une transclusion, et en placer le résultat comme enfant d'un autre objet. Le parent va alors contenir une référence (sous forme de créneau) identifiant la transclusion d'une part et la source d'autre part.

- 10 Lorsque l'utilisateur modifie le résultat d'une transclusion, étant donné que ce résultat n'a pas d'existence propre, les modifications seront stockées « dans » la transclusion elle-même (en terme de différences).

- 15 Une transclusion est en effet traitée, du point de vue de la mise en œuvre du procédé présenté ici, comme un objet dans le sens où elle peut être modifiée dans les différentes versions et que ces modifications peuvent être refusées par d'autres utilisateurs, ce qui fait que la transclusion peut avoir plusieurs branches. Le contenu d'une transclusion décrit les différences appliquées aux sources. Une transclusion diffère d'un objet par le fait qu'une transclusion ne peut elle-même être placée comme enfant d'un objet, et ne peut directement être affichée à l'utilisateur (il ne peut que voir l'image d'objets par cette transclusion).

20 Mise en oeuvre

Architecture

Les ressources identifiables par un ERC donné contiennent des objets éventuellement définis comme étant *remplaçables*²³ dans le cadre de cet ensemble.

- 25 Par le procédé décrit ici, les utilisateurs d'un même ERC peuvent modifier et contribuer ainsi de nouvelles versions de certains de leurs objets remplaçables. Celles-ci vont remplacer²⁴ celles présentées aux utilisateurs de cet ensemble, ou les enrichir en apportant (en option) des versions alternative²⁵s représentant des « points de vue différents ». C'est le serveur collaboratif (ou dispositif analogue) qui détermine quels sont ces objets.

- 30 Une architecture simple (parmi d'autres) pour mettre en œuvre ce procédé est retenue comme base de description du procédé de l'invention. Elle est présentée, très schématiquement, à la figure 7 et comprend les éléments suivants :

- Outil de manipulation d'information (représenté par l'icône d'un ordinateur portable) : Moyen de présentation et/ou de manipulation de ressource. Peut notamment être un navigateur tel qu'Internet Explorer, ou un outil tel que Microsoft Word, ou encore
35 l'explorateur de fichiers du système d'exploitation. En pratique on dotera cet outil d'une extension (plug-in) apte à mettre en œuvre le procédé.
- Serveur de contenu primaire (PCS) : Serveur apte à fournir la version de ressource requise par l'utilisateur dans l'outil de manipulation d'information.
- Serveur collaboratif (ES) : Serveur apte à fournir, pour une version de ressource
40 accédée, les identifiants des versions d'objets.

²³ Dans la suite, tous les objets de toutes les ressources sont considérés comme remplaçables (toutefois on ne considérera comme objet que ceux qui sont réifiés), mais d'autres méthodes peuvent être mises en œuvre pour décider sélectivement ceux qui le sont. En particulier, on met en œuvre des contraintes de non-modifiabilité de certains objets (qui par exemple sont des publicités) comme expliqué plus loin.

²⁴ (remplacer soit leur contenu, soit leur position, soit les deux, ou encore en supprimer ou ajouter).

²⁵ Comme déjà indiqué, une autre architecture est décrite à la fin de ce document.

- Serveurs de contenus secondaire
versions d'objets.

22

seurs aptes à fournir les contenus des

Comme l'illustre schématiquement la figure 7, l'utilisateur ayant requis une version de ressource en donnant son identifiant (URL, URI, ...) selon des moyens classiques, et pour
5 chaque identifiant d'ERC donné par l'utilisateur²⁶, l'outil de manipulation d'information effectue les étapes suivantes :

1. obtient la version de ressource requise d'un serveur de contenu primaire ;
2. (en parallèle avec l'étape 1.) obtient du serveur collaboratif les identifiants des versions d'objets à présenter ;
- 10 3. (suite à l'étape 2.) obtient des serveurs de contenus secondaires lesdites versions d'objets ;
4. et assemble lesdites versions d'objets dans ladite version de ressource demandée.

Nous allons à présent décrire le fonctionnement d'un serveur collaboratif. Le terme "le système" indique toujours ci-après le serveur collaboratif.

15 Structures de données

Il y a essentiellement quatre types de structures de données qui sont manipulés par le système, qui représentent :

1. Une ressource collaborative
- 20 2. Un objet (d'une ressource collaborative) - ceci représente non pas une version particulière de cet objet mais la classe de toutes ses branches et versions (les exemples utilisent les lettres a, b, c, etc pour représenter un objet). Comme dit plus haut il peut s'agir d'une transclusion et non d'un objet.
3. Une version (d'une ressource collaborative)
- 25 4. Une référence à un contenu (i.e. d'une version particulière d'une branche d'un objet) qui se trouve dans un serveur de contenus.

On va décrire quelles sont les données stockées et tenues à jour dans ces structures de données.

Un *objet* possède

- un identifiant unique pour la ressource collaborative,
- une liste de branches et pour chacune un identifiant (il n'est pas nécessaire que les
30 identifiants d'objets ou de branches soient uniques entre différentes ressources collaboratives).
- Afin d'éviter d'avoir à chercher à tout moment la version la plus récente, il est avantageux de stocker pour chaque branche une référence vers la version (structure de version de ressource collaborative) la plus récente de son contenu. (Il est aussi possible de se passer
35 de cette information et d'aller à chaque requête parcourir la table des versions et consulter les dates des modifications des versions locales).

Une *référence à un contenu* indique

- l'identifiant et le serveur à partir desquels les données effectives peuvent être obtenues
- quels sont les objets que le contenu référence via des créneaux
- 40 • la date à laquelle ce contenu a été modifié la dernière fois

²⁶ L'utilisateur peut activer zéro, un ou plusieurs ERC au moment de sa requête. Ces ensembles peuvent aussi être configurés par défaut dans l'outil de manipulation d'information.

- Le cas échéant, la ou les catégories de ce contenu.

Une *version* spécifique

- le client qui en est le propriétaire
 - le cas échéant, de quelle autre version de la ressource collaborative elle dérive
- 5 • pour n'importe quel objet (donné par son identifiant) elle peut indiquer quelle branche est activée (et ne spécifie rien pour les objets pour lesquels il faut récupérer le choix de branche donné par la ressource collaborative dont elle dérive)
- pour n'importe quelle branche de chaque objet pour laquelle elle a fourni une version locale, une référence au contenu de cette version locale.
- 10 Enfin, une *ressource collaborative* regroupe tout ceci, en stockant
- l'ensemble des objets qu'elle contient
 - l'identifiant de l'objet racine de la ressource collaborative (qui est donc commun à toutes les versions)
 - l'ensemble des versions de la ressource collaborative

15 Contraintes

Ainsi, les algorithmes décrits ci-après interrogent les contraintes (que tous les objets doivent être uniques, par exemple) en demandant si une certaine combinaison de branches d'une ressource collaborative doit être autorisée ou non. Cette façon de faire permet de laisser ouverts les types de contraintes pouvant être appliqués.

- 20 Afin d'aider à la satisfaction des contraintes, ces dernières, au lieu de simplement refuser en bloc une certaine combinaison de branches, la mise en oeuvre de la contrainte peut fournir une liste des branches posant problème. Par exemple pour une contrainte d'unicité, il s'agit de la liste des branches contenant l'objet multiplié.

Algorithmes

- 25 Les sections suivantes vont décrire les différentes opérations qu'un serveur collaboratif offre à son environnement, qui sont :
- Accès en "lecture" (construction de la structure à présenter pour une version d'une ressource collaborative)
 - Changement de branche active dans une version de la ressource collaborative.
- 30 • Modification d'un objet
- Création d'un objet
 - Refus d'un objet

Construction de la structure à présenter pour une version de la ressource collaborative

- 35 À partir des identifiants d'une ressource collaborative et d'une version particulière de cette ressource collaborative, on donne les références aux contenus des objets devant être fournis au client afin que celui-ci puisse construire la ressource collaborative telle qu'elle doit être présentée au client. (On rappelle que le serveur collaboratif n'accède pas aux contenus eux-mêmes mais seulement à leurs identifiants, ce qui est la raison pour laquelle il ne peut lui-même construire la page).
- 40

On part de la racine de la ressource collaborative et on détermine quelle en est la branche active dans la version demandée. Dans le cas où il s'agit d'un objet vu à travers transclusion il faut à ce moment appliquer les différences de la transclusion (comme décrit dans la partie de ce document sur les transclusions).

- 5 Puis on répète cette opération à partir des enfants de ce contenu.

Changement de branche active

Le client demande un changement de branche active en indiquant quel objet de quelle version de ressource collaborative est affectée, ainsi que l'identifiant de la nouvelle branche choisie.

- 10 Le système vérifie tout d'abord si la combinaison de choix de branches résultante pour cette version est en violation d'une contrainte. Si c'est le cas il va automatiquement changer quelles sont les branches actives dans les autres objets pour satisfaire les contraintes.

Une fois une combinaison valide trouvée, le système va simplement stocker la nouvelle valeur dans la structure.

Réaction à une modification

- 15 Cette section décrit comment le système est informé d'une modification, et comment il tient à jour ses structures de données pour tenir compte de cette modification.

- 20 La description d'une modification contient un identifiant de ressource collaborative, un identifiant de version (de la ressource), l'identifiant de l'objet modifié et la description du contenu de l'objet après la modification (tous les attributs énumérés dans la description de la structure de données pour "référence à un contenu").

À noter qu'en pratique on va grouper un ensemble de modifications dans une seule requête, et traiter ces modifications en parallèle. En particulier il faut vérifier les contraintes seulement après avoir effectué toutes les modifications et non entre chacune d'entre elles.

- 25 Étant donné qu'une modification peut changer l'effet des contraintes, il se peut que d'autres versions se retrouvent en violation d'une contrainte. Si ceci se produit le procédé va effectuer un refus des branches posant problème (ceci revient à simuler l'utilisateur qui refuse). Ceci aura pour effet que les versions ayant activé ces branches conserveront les anciennes valeurs des objets.

- 30 Il peut arriver qu'une modification soit par elle-même en violation d'une contrainte (par exemple si l'utilisateur tente de placer dans un même objet deux fois le même enfant soumis à contrainte d'unicité. À ce moment toute combinaison de branches ayant choisi cette branche sera en violation de la contrainte. Il peut être préférable dans ces cas que simplement la modification échoue, plutôt qu'elle soit acceptée et ensuite que le système refuse de l'afficher.

- 35 Après avoir éventuellement créé ces nouvelles branches il suffit de placer les nouvelles informations (contenus et listes d'enfants) dans la branche originale.

On met également à jour dans la ressource collaborative le pointeur vers la version la plus récente dans cette branche.

Traitement d'une création d'objet

- 40 Comme déjà expliqué précédemment, une création d'objet est effectuée en deux étapes. Tout d'abord la création elle-même, puis la modification de l'objet parent pour contenir une référence vers cet objet. La première étape est décrite ci-après, et la seconde étape sera demandée par le client au système comme s'il s'agissait d'une modification "normale".

Une demande de création d'objet indique la source collaborative cet objet est créé, quel doit être le contenu initial (sous forme d'un identifiant) et quelle version de la ressource collaborative a effectué la création. La réponse à cette demande sera l'identifiant du nouvel objet.

5 Le traitement de cette requête consiste à :

- choisir des nouveaux identifiants d'objet et de branche
- créer une nouvelle structure de contenu avec l'identifiant donné, dont la date est maintenant et qui ne possède pas d'enfants
- créer une nouvelle structure d'objet avec le nouvel identifiant choisi et une seule branche dont la version la plus récente est celle qui a effectué la création
- ajouter le nouvel objet à la liste des objets de la ressource collaborative
- insérer l'identifiant qui donne le contenu initial donné dans la version.

10

On fait l'hypothèse que le contenu initial de l'objet est vide ou neutre du point de vue des contraintes et donc qu'il n'y a rien à vérifier (dans le cas contraire on pourrait par exemple imaginer une contrainte fixant un maximum à la taille totale, par exemple en nombre de caractères, de la ressource collaborative. Dans ce cas la création d'un objet initialisé d'un contenu non nul aurait pu violer la contrainte).

15

En effet, si l'utilisateur crée un objet non vide, ceci doit être envoyé au système en deux étapes, d'abord la création d'un objet vide puis sa modification vers un objet non vide.

20 **Traitement d'un refus**

Pour notifier d'un refus, le client informe le système quelle version de ressource collaborative refuse quels objets. On part du principe que les versions refusées sont celles qui sont affichées pour le moment aux emplacements des objets pour la version donnée.

25

Il peut arriver qu'un refus échoue, par exemple au cas où l'utilisateur cherche à refuser l'unique version d'un objet dans une branche. À ce moment l'erreur est signalée à l'utilisateur et le système ne fait rien.

On détermine tout d'abord quelles versions de la ressource collaborative fournissaient les contenus refusés. Pour ce faire on regarde pour chaque branche refusée quelle est la version de l'objet la plus récente pour cette branche.

30

Au cas où le refus causerait une violation de contrainte, le système va tenter de résoudre le problème en refusant également une autre branche posant problème. Ceci peut se produire lorsqu'un utilisateur veut refuser un déplacement. En cas de contrainte de non-multiplicité (i.e. il doit y avoir au plus une instance d'un objet donné), si l'utilisateur refuse la disparition de l'objet à la source du déplacement, le système l'oblige à également refuser l'apparition de l'objet à la destination.

35

On crée un nouvel identifiant de branche pour l'objet refusé. On déplace le contenu refusé dans la nouvelle branche, pour la version de la ressource collaborative qui avait fourni ce contenu. Cette opération a aussi comme effet d'enlever le contenu en question pour l'ancienne branche dans cette version.

40

Étant donné que la version supprimée était la plus récente il faut maintenant recalculer quelle est la version la plus récente dans cette branche.

Dans le cas où le contenu refusé était actif dans la version fournissant ce contenu on rend la nouvelle branche active dans cette version.

26
Finalement, au cas où la version d t de la version refusée, on active (explicitement) l'ancienne branche dans cette dernière (afin d'éviter qu'elle suive la version refusée dans la nouvelle branche).

Gestion des opérations concurrentes

- 5 Le système décrit jusqu'à présent suppose que les modifications effectuées par les utilisateurs sont atomiques, c'est-à-dire d'une part qu'elles sont basées sur l'état le plus récent de la ressource collaborative et d'autre part que deux utilisateurs ne tentent pas de modifier simultanément le même objet. Ceci ne peut pas être garanti en réalité, par exemple si un utilisateur veut modifier un objet et qu'un autre veut simultanément le refuser.
- 10 En particulier, les actions effectuées par l'utilisateur risquent d'être faites sur la base d'une version obsolète d'une ressource collaborative, celle-ci ayant été modifiée depuis la dernière fois qu'il l'a téléchargée sur son client.

Voici deux méthodes pour résoudre ce problème.

- 15 On utilise un système de verrous, où un utilisateur souhaitant modifier un objet doit tout d'abord le verrouiller et en prendre les droits exclusifs le temps de sa modification, intervalle de temps pendant lequel personne n'est autorisé à modifier l'objet.

On ne prend pas de mesures préventives mais si un utilisateur cherche à effectuer une opération qui est ainsi en conflit, on lui demande d'effectuer lui-même la fusion des deux opérations conflictuelles.

- 20 Par exemple si deux utilisateurs téléchargent une ressource collaborative, chacun y modifie une partie, et chacun tente d'envoyer sa version, au moment où le système reçoit la deuxième requête de modification il va renvoyer à l'utilisateur la version à jour de la ressource collaborative et lui demander d'y appliquer à nouveau ses modifications.

- 25 Le système peut fournir à l'utilisateur des outils pour effectuer cette fusion rapidement : si le système détecte la nature des modifications faites par les utilisateurs il peut tenter de fusionner les actions des deux utilisateurs automatiquement, et demander l'aide du dernier uniquement lorsqu'il s'agit de prendre des décisions, e.g. si la même partie d'un paragraphe a été modifiée par chaque utilisateur.

Restrictions de propagation

- 30 On peut offrir une fonction pour demander, objet par objet, des variations par rapport au comportement normal des ressources collaboratives.

Protection en lecture

- 35 Un utilisateur peut demander à avoir pour un objet une branche "secrète" à laquelle il est le seul à avoir accès. Une telle branche se comporte en tout point comme une autre à la différence qu'elle n'est pas rendue visible aux autres utilisateurs (et ils n'ont pas la possibilité de la sélectionner). Si un utilisateur crée un objet comme enfant d'une telle branche secrète, le nouvel objet sera également rendu secret.

On peut bien sûr envisager des réglages plus fins où un groupe non réduit à l'utilisateur seul est autorisé à accéder à cette branche.

- 40 **Protection en écriture**

L'utilisateur peut demander à toujours voir sa version locale pour objet particulier, même si une version plus récente existe dans la même branche. Une façon de mettre ceci en œuvre est que sitôt qu'une modification a lieu dans une branche, un tel objet protégé en écriture est

27
envoyé dans sa propre branche (comme ²⁷ sé). Si l'utilisateur modifie ensuite cet objet à nouveau, il est à nouveau envoyé dans la branche où il se trouvait précédemment, de sorte que cette nouvelle version est rendue visible aux autres utilisateurs.

- 5 Lorsque l'objet est ainsi déplacé dans sa propre branche l'utilisateur a toujours la possibilité de consulter l'autre branche afin d'y voir la dernière version.

D'autre part les versions dérivant de celle contenant le contenu protégé en écriture continueront à voir le contenu protégé en question, comme elles héritent de ses branches actives (sauf bien sûr si elles ont explicitement activé une autre branche).

Propagations unidirectionnelles

- 10 Comme autre type de restriction on peut avoir un objet ou une ressource pour lequel les modifications se propagent uniquement dans un seul sens, par exemple de l'amont vers l'aval comme c'est le cas de la transclusion.

Variante : Versions de ressources comme Transclusions

- 15 La transclusion peut davantage être utilisée pour collaborer (en propageant les modifications les plus récentes ou les plus élaborées) si ce mécanisme est étendu pour également propager les modifications en direction de la source (vers l'amont). On va maintenant décrire cette généralisation de la transclusion (pour servir de mécanisme de collaboration telle que définie jusqu'ici).

- 20 Au lieu d'avoir des versions différentes d'une même ressource collaborative on va maintenant considérer que chaque telle version est en fait une ressource (aval) à part entière (ayant un identifiant de ressource distinct) qui transclut tout son contenu de la ressource (amont) qu'elle a pour but de modifier. On dit que la ressource aval *dérive* de la ressource amont.

- 25 De même, au lieu d'avoir des versions d'objets on va maintenant considérer des objets à part entière (ayant des identifiants d'objets distincts) qui résultent d'une transclusion à partir des objets correspondants de la ressource amont²⁷. Le procédé consiste ainsi essentiellement déterminer quels objets doivent être remplacés (par un objet de remplacement) ou enrichies (par des objets alternatifs) lors de la présentation d'une ressource à l'utilisateur.

- 30 Comme tant la source (amont) que la destination (aval) ont maintenant des identifiants d'objets distincts, il est possible de les considérer en parallèle. Par exemple d'avoir dans un site une page qui est transclue d'une autre ou d'avoir dans un répertoire une ressource transclue d'une autre se trouvant dans le même répertoire.

- 35 Les nouveaux procédés de collaboration que l'on va maintenant décrire ont été conçus pour s'intégrer (et être complémentaire) avec le procédé de transclusion. La différence essentielle est que les sens amont-aval et aval-amont de collaboration ne seront maintenant plus symétriques. On va ainsi décrire ici des objets « les plus élaborés » au lieu de versions « les plus récentes » d'objet.

Illustration du principe du procédé

Les figures 8 à 21 illustrent différentes étapes d'utilisation du procédé et permettent de le comprendre intuitivement, avant même de lire sa description qui va suivre.

- 40 Ces figures illustrent des exemples de dérivation de ressource : une ressource R1 est dérivée en une ressource R2 afin d'être élaborée davantage, puis R2 est elle-même dérivée pour y incorporer encore d'autres modifications, et ainsi de suite. Cependant un objet ainsi modifié dans une ressource peut être refusé par un des propriétaires des ressources dont elle est issue,

²⁷ Et on va adapter la mise en œuvre du concept de « branche » en conséquence.

et donne alors lieu à la création d'une autre branche représentent ainsi des points de vue différents. Dans chaque branche, l'objet qui est le plus à l'aval est « le plus élaboré ».

Situation initiale (figure 8): La ressource R1 a l'objet a1. Donc a1 est le plus élaboré.

R2 dérive de R1 et modifie a1 en a2 (figure 9): Donc a2 est le plus élaboré.

R3 dérive de R2 et modifie a2 en a3 (figure 10): Donc a3 est le plus élaboré.

R2 refuse a3 (figure 11): En résultat de ce refus, une nouvelle branche est créée (Branche 1) alors que le flux original est dans Branche 0. a2 est le plus élaboré dans Branche 0 et a3 est le plus élaboré dans Branche 1.

Le diagramme de branches résultant est présenté à la figure 12.

Lorsque l'utilisateur visite R1 ou R2 il voit ce qui est montré à la figure 13.

En revanche, lorsqu'il visite R3, il voit ce qui est présenté à la figure 14.

R4 dérive de R2 et modifie a2 en a4 (figure 15): Donc a4 est le plus élaboré dans Branche0.

Lorsque l'utilisateur visite R1 ou R2 ou R4, il voit ce qui est présenté à la figure 16.

En revanche, lorsque l'utilisateur visite R3, il voit ce qui est présenté à la figure 17.

R2 refuse a4 (figure 18): Une nouvelle branche est créée (Branche 2) avec a4 comme objet le plus élaboré.

Le diagramme de branches est maintenant comme dans la figure 19.

On remarque que lorsque l'utilisateur voit un objet dans Branche 2, par exemple dans la ressource R4, les objets alternatifs ne sont pas tous sur un pied d'égalité, Branche 0 est plus pertinente que Branche 1. On pourrait limiter la visibilité d'une branche aux branches ancêtres et descendantes, ou encore trier les objets alternatifs selon la proximité de leur branche.

R5 dérive de R2 en modifiant a2 en a5 (figure 20):²⁸ Donc a5 est l'objet le plus élaboré dans Branche 0.

R1 accepte a3 (figure 21): En résultat de cette acceptation, a1 est remplacé par a3 et mis dans Branche 1.

Procédé de création d'objets les plus élaborés

On décrit un procédé de modification de ressource comprenant les deux grandes étapes suivantes :

1. dérivation de la ressource dont au moins un objet est à être modifié, et modification de cet (ou ces) objet(s) dans la ressource dérivée,

2. proposition de cette modification à l'amont et détermination de la position des objets « les plus élaborés » en fonction de la réponse (ou la non réponse) à cette proposition.

Ces deux grandes étapes sont respectivement décrites dans les deux sections suivantes. On va ensuite décrire comment on détermine les objets de remplacement (OR) et les objets alternatifs (OA).

Création de ressource dérivée

Règle 0 : Lorsque l'utilisateur accède dans le cadre d'un ERC à une ressource au moyen d'un outil de manipulation d'information, en vue de la modifier, mais qu'il ne peut ou ne veut la

²⁸ (ou dérive de R1, ou encore de R3, R4 ou R5 mais choisit l'alternative a2 puis la modifie)

modifier directement²⁹, l'outil de man²⁹ formation crée automatiquement une *ressource dérivée* qui présente le même contenu³⁰ et la présente à l'utilisateur, et ce dernier modifie ladite ressource dérivée à la place de ladite ressource accédée.

La ressource dérivée est stockée dans un serveur quelconque³¹ disponible à l'utilisateur (et donc pas forcément dans le serveur de ladite ressource accédée).

Les modifications sont proposées à l'amont

On dit que ladite ressource accédée en vue d'être modifiée est à *l'amont* de ladite ressource dérivée (et que cette dernière est donc à *l'aval* de ladite ressource accédée). En effet, une ressource peut être dérivée d'une ressource elle-même dérivée, formant ainsi une chaîne (ou une cascade) amont-aval.

La situation initiale est régie par la règle suivante.

Règle ½ : Les objets contenus dans une ressource qui d'une part n'est pas dérivée d'une autre et d'autre part à partir de laquelle il n'existe pas de ressource dérivée sont (implicitement) considérés comme « les plus élaborées ».

On a ensuite les règles suivantes :

Règle 1 : Chaque objet modifié³² dans une ressource dérivée, tant qu'il n'a pas été modifié plus à l'aval, est *le plus élaboré* (et l'objet le plus proche le cas échéant à l'amont cesse de l'être).

Règle 2 : Ladite modification est spontanément proposée aux propriétaires³³ respectifs des ressources correspondantes à l'amont.

Chacun de ceux-ci peut *accepter*, *refuser* ou *ignorer* ladite proposition. En voici les conséquences :

- **Règle 3 :** Dans le cas où la modification d'un objet est *acceptée*, le résultat de cette modification est propagé à l'objet acceptant.
- **Règle 4 :** Dans le cas où la modification d'un objet est *refusée* à l'amont, quelque soit la position de l'objet refusant, le premier objet qui se trouve à l'amont (dudit objet modifié refusé)
 - devient *le plus élaboré*³⁴
 - et en commençant par cet objet refusé une branche distincte est créée (dans l'arbre des modifications d'objets) pour y placer l'identifiant dudit objet modifié refusé³⁵ ainsi que ses descendants. Il peut ainsi y avoir plusieurs objets marqués « le plus

²⁹ (soit parce qu'elle est en lecture seule ; soit on est dans le cas de l'utilisateur qui cherche à modifier une ressource qu'il a le droit de modifier mais qu'il ne veut pas modifier directement car il préfère plutôt modifier une ressource dérivée d'elle)

³⁰ Ceci est décrit à la section « Procédé de Dérivation par Transclusion ». A noter que le contenu de la ressource dérivée reste ensuite synchronisé avec la ressource dont elle dérive.

³¹ (choisi par l'utilisateur)

³² Il peut notamment s'agir de l'ajout d'un objet, de la suppression d'un objet, ou encore de la modification du contenu ou de la position d'un objet. Alternativement, on peut adopter l'approche des « crêneaux » décrite plus loin en détail.

³³ Les modifications peuvent être proposées seulement si ledit propriétaire est accessible, par exemple lorsque son adresse email est connue. Quand ladite ressource accédée est elle-même une ressource dérivée (dans le cadre d'un ERC), le problème ne se pose pas puisque son propriétaire est déjà connu et accessible d'office. En fait, le procédé de remplacement/enrichissement d'objet pourra lui-même servir de moyen de proposition d'acceptation ou non de la modification (de l'objet qui justement sert de remplacement ou d'enrichissement).

³⁴ (sans que les objets qui étaient les plus élaborés avant ce refus cessent de l'être)

³⁵ (de manière à pouvoir par la suite prolonger le « tronc », sans suivre cette branche)

élaboré » dans un arbre de mo
sont des *objets alternatifs*.

cernant un même objet³⁶; on dit que ce

Exemple

On va maintenant expliquer et illustrer (voir figure 22) ces règles en considérant deux
5 dérivations (et modifications associées) successives à partir d'une ressource R1.

Noter que ces dérivations peuvent être effectuées par des utilisateurs différents³⁷.

Situation initiale de l'exemple

Les ressources sont identifiées par une lettre en majuscule suivie d'un chiffre (indice) qui
10 indique leur numéro de version ; les objets sont identifiés en minuscule et ont le même indice
que la ressource dans laquelle ils ont été créés ou modifiés.³⁸

La figure 22 illustre que :

- Le contenu de la ressource R1 est l'objet ***r1***. *r1* contient lui-même trois objets : ***a1***, ***b1***
15 et ***c1*** (en réalité, l'utilisateur qui voit le contenu de la ressource R1 voit l'objet *r* qui
contient les objets *a*, *b*, *c* ; les indices nous serviront ici à voir plus facilement quelle est
la ressource qui contenait l'objet en question initialement)
- R1 est dérivée (en R2) pour modifier le contenu spécifique³⁹ ou la position⁴⁰ des objets
20 *r*, *a* et *c* (*r1* est modifié en ***r2***, *a1* est modifié en ***a2*** et *c1* est modifié en ***c2***) et insérer un
nouvel objet *d* (***d2***).
- R2 à son tour est dérivée (en R3) pour modifier *b* (*b2* est modifié en ***b3***) et supprimer *c*
(supprimer *c2*).

Dans la mise en œuvre du serveur collaboratif que nous allons maintenant décrire, la structure
de données gérée dans le serveur collaboratif ne contient avantageusement pas les ressources
ou les objets eux-mêmes, mais seulement leurs identifiants respectifs. On présente ici de
manière schématique une structure de pointeurs et marqueurs, les pointeurs permettant de
25 reconstituer les dérivations et modifications et les marqueurs servant à distinguer les objets les
plus élaborés (ainsi que les suppressions d'objet). Cette structure comprend :

- Des arbres de dérivation de ressources (tel que R1→R2→R3)
- Des arbres de modification d'objets⁴¹ (tels que : *r1*→*r2* ; *a1*→*a2* ; *b1*→*b3* ; *c1*→
30 *c2*→*e3* ; *d2*) dont les nœuds sont accessibles à partir des nœuds correspondants des
arbres de dérivation de ressources⁴².

Avantageusement, en réalité les nœuds formant les arbres de modification d'objets ne
comprennent que les objets modifiés. Les nœuds manquants représentent des objets non

³⁶ En effet, il y a un objet le plus élaboré différent dans chaque branche différente.

³⁷ (différents entre eux et/ou différents du propriétaire de R1)

³⁸ Pour plus de clarté, on met en gras et italique les objets créés ou modifiés.

³⁹ Par exemple, s'il s'agit d'éléments dans une notation à la XML, le contenu de *r1* et *a1* sont leurs « innertext ».

⁴⁰ La position peut être spécifiée par l'objet parent et un objet frère (à gauche ou à droite, selon la convention prise). On va décrire une variante à cette approche plus loin à la section « Créneaux » (à noter que l'approche des créneaux était celle adoptée dans le chapitre précédent).

⁴¹ (ou « arbres de dérivations d'objets » - noter que plus loin on considère des objets composites appelés « grains » et on manipule alors des « arbres de modification de grains »)

⁴² Chaque nœud de l'arbre de dérivation de ressources pointe sur les nœuds correspondants des arbres de modification d'objets (comme on le voit dans les figures). En pratique, on peut mettre en œuvre une structure de données différente mais équivalente, notamment en utilisant des tables de BD relationnelle.

modifiés, c'est-à-dire qu'ils reflètent⁴³ le contenu de leurs correspondants à l'amont.⁴⁴

31

Les figures présentées montrent schématiquement un arbre de dérivation de ressources et des arbres de modification d'objets. Elles montrent, sous forme de rectangle noir dans les arbres de modification d'objets, les nœuds marqués comme étant *les plus élaborés*.

La figure 23 présente la structure de données avant les dérivations (c'est-à-dire avant que ni R2 ni R3 n'existe). Les objets r1, a1, b1 et c1 sont marqués puisqu'en vertu de la Règle 1/2 ils sont les plus élaborés.⁴⁵

La figure 24 présente la structure de données après les modifications dans R2 et avant la dérivation de R3. Elle montre les nœuds marqués comme étant les plus élaborés (i.e. les rectangles noirs) en supposant qu'aucune modification n'a encore été acceptée ou refusée. On voit dans la figure qu'en vertu de la Règle 1, les objets r2, a2, c2 et d2 deviennent les plus élaborés, tandis que b1 reste le plus élaboré, puisque b2⁴⁶ n'a pas été modifié.

La figure 25 présente la structure de données après les modifications dans R3. Ici encore les nœuds marqués comme étant les plus élaborés présupposent qu'aucune modification n'a encore été acceptée ou refusée. Rappelons qu'une des modifications effectuées a consisté à supprimer c. On remarque ainsi que l'objet c2 n'est plus le plus élaboré mais que c'est e3 (l'absence de l'objet c) qui l'est (toujours en vertu de la Règle 1).

Rappelons de nouveau que dans R2 il y a bien l'objet b2 (qui reflète le contenu et position de b1) et que dans R3 il y a bien les objets r3, a3 et d3 (qui reflètent les contenus et positions de r2, a2 et d2 respectivement), mais qu'ils ne figurent pas dans les arbres de modification de ces objets puisqu'on n'y stocke que les nœuds représentant les objets modifiés.

La figure 25 représente la situation initiale sur laquelle se basent les exemples qui suivent. Ainsi, on va considérer qu'avant la dérivation (puis modification) de R2 en R3, il n'y a eu ni acceptation ni refus d'un quelconque objet.

Procédé de détermination de l'objet de remplacement et des objets alternatifs

Règle 5 : Chaque objet courant devant être présenté à l'utilisateur, si lui-même n'est pas le plus élaboré, est remplacé (au sein de ladite présentation) par son correspondant le plus élaboré se trouvant à l'aval dans la même branche.

Bien évidemment, l'utilisateur peut à tout moment activer et désactiver cette fonctionnalité de remplacement d'objets.

Règle 6 : Les objets « les plus élaborés », sauf l'objet qui sert d'*objet de remplacement* en application de la Règle 5, sont considérés comme étant des *objets alternatifs* et sont présentés à l'utilisateur comme des options. Chaque objet alternatif a une priorité en fonction de la

⁴³ (grâce au procédé de transclusion, décrit plus loin)

⁴⁴ Précision : Selon la méthode utilisée, bien qu'en principe, dans les arbres de modification d'objets, ne sont représentés que les identifiants des objets dont le contenu ou la position est modifié (ou encore les suppression d'objet), on peut aussi y mettre les nœuds représentant des objets non modifiés mais qui sont néanmoins marqués comme étant les plus élaborés. Il existe en effet plusieurs mises en œuvre possibles d'un mécanisme de gestion d'objets les plus élaborés et la méthode présentée ici est l'une seulement de celles-ci.

⁴⁵ Dans une mise en œuvre optimisée, on peut éviter de représenter dans les objets non modifiés. On les présente ici pour des raisons de clarté.

⁴⁶ b2 n'est pas dans l'arbre de modification des objets puisque cet arbre ne comprend que les identifiants des objets modifiés (sauf ceux non modifiés mais étant le plus élaboré) comme déjà mentionné.

proximité de la branche dans laquelle il ³² objet se trouvant dans la même branche étant prioritaire et les branches plus proches ayant une priorité plus forte).⁴⁷

Lorsque l'utilisateur sélectionne un objet alternatif, ce dernier remplace l'objet courant affiché (ce dernier pouvant être l'objet contenu dans la ressource, un objet de remplacement ou un autre objet alternatif sélectionné précédemment).

Il est avantageux pour l'utilisateur de lui présenter ensemble tous les objets alternatifs provenant d'une même ressource, ceci pour des raisons de cohérence. Lorsque l'utilisateur visionne une ressource, on groupe les objets alternatifs d'après les ressources où ils se trouvent.

Bien évidemment, l'utilisateur peut à tout moment activer et désactiver cette fonctionnalité de présentation d'objets alternatifs.

La mise en œuvre peut simplement se faire de la manière suivante. Pour déterminer les objets de remplacement et alternatifs le serveur collaboratif retrouve, dans l'ensemble des arbres de dérivation de ressources, le nœud représentant la ressource dérivée en question. A partir de là il retrouve pour chaque objet pour lequel il existe un arbre de modification d'objets, en parcourant cet arbre, l'objet de remplacement – l'objet le plus élaboré dans la même branche à l'aval – et les objets alternatifs (les autres objets les plus élaborés dans les autres branches) selon les Règles 5 et 6.

20 Exemples

Ignorance ou acceptation (fig. 26)

Supposons que l'utilisateur visite R1 avant tout acceptation ou refus. Lors de la présentation de R1 à l'utilisateur (figure 26), en vertu de la Règle 5, r2, a2, b3, l'absence de c, et d2 vont remplacer les objets correspondants de R1. Quant à la Règle 6, il n'y a pas d'objet alternatif à présenter puisqu'il n'y a chaque fois qu'un seul objet le plus élaboré et celui-ci est déjà utilisé comme objet de remplacement au sens de la Règle 5.

En conséquence des règles énoncées, tant qu'il n'y a pas eu de refus, la ressource et ses dérivées sont présentées de la même façon y compris après des acceptations, (sauf s'il y a eu des modifications à l'amont postérieures à des modifications à l'aval, auquel cas on doit se référer à la règle 7 énoncée plus loin).

Refus (fig. 27 à 32)

Supposons que l'utilisateur visite R1 après que « R2 refuse ». Lors de la présentation de R1 à l'utilisateur, en vertu de la Règle 5, r2, a2, b2, c2 et d2 vont remplacer les objets correspondants de R1. En outre, par application de la Règle 6, les objets b3 et l'absence de c sont présentés comme option par rapport aux objets correspondants qui sont présentés dans la ressource R1 (voir figure 27).

Supposons que l'utilisateur visite R1 après que « R1 refuse R3 et d2 ». Lors de la présentation de R1 à l'utilisateur, en vertu de la Règle 5, r2, a2, b2 et c2 vont venir remplacer r1, a1, b1 et c1. En outre, par application de la Règle 6, les objets b3, l'absence de c, et d2 seront présentés comme options par rapport à leurs objets (ou absence d'objet) correspondants qui sont présentés dans la ressource R1 (voir figure 28).

⁴⁷ Une approche supplémentaire de priorisation des objets alternatifs est décrite plus loin à la section « Procédé de tri d'objets alternatifs par notation (rating) ».

Supposons que l'utilisateur visite R2 après que « R1 refuse R3 et d2 » (voir figure 15). Il voit les objets de R3 qui reflètent en fait les objets de R2, sauf pour b3 et c3 qui ont été remodifiés dans R3. Il n'y a pas de remplacement puisque tous ces objets sont déjà les plus élaborés. Cependant, en vertu de la Règle 6, b2 et c2 représentent des objets alternatifs par rapport respectivement à b3 et à l'absence de c (voir la figure 31).

5 Supposons que l'utilisateur visite R2 après que « R1 refuse R3 et d2 ». Il n'y a pas de remplacement au sens de la Règle 5 puisque les objets de R2 (ou reflété dans R2) sont eux-mêmes les plus élaborés. Il y a les objets alternatifs b3, l'absence de c dans R3, et l'absence de d dans R1, au sens de la Règle 6 (voir la figure 30).

10 Supposons que l'utilisateur visite R3 après que « R2 refuse ». Il voit les objets de R3 qui reflètent en fait les objets de R2, sauf pour b3 et c3 qui ont été remodifiés dans R3. Il n'y a pas de remplacement puisque tous ces objets sont déjà les plus élaborés. Cependant, en vertu de la Règle 6, b2 et c2 représentent des objets alternatifs par rapport respectivement à b3 et à l'absence de c (voir la figure 31).

15 Supposons que l'utilisateur visite R3 après que « R1 refuse R3 et d2 » (voir figure 15). Il voit les objets de R3 qui reflètent en fait les objets de R2, sauf pour b3 et l'absence de c qui sont des modifications faites dans R3. Il n'y a pas de remplacement puisque tous ces objets sont déjà les plus élaborés. Cependant, par rapport à la Règle 6, b2, c2 et l'absence de d représentent des objets alternatifs (voir figure 22).

Dérivations multiples d'une même ressource (fig. 33 à 44)

20 Supposons que l'utilisateur accède à la ressource R1 juste après que « R1 refuse R3 et d2 » puis veuille modifier les objets a2 et b2 (de remplacement) qui lui sont présentés d'office. En vertu de la Règle 0 une ressource R4 dérivée de R1 se crée automatiquement et ces modifications engendrent les objets a4 et b4. La figure 33 illustre le fait que a4 succède à a2 et non à a1, car a2 est l'objet de remplacement de a1 (c'est donc a2 qui est modifié en a4). Par ailleurs, b4 prolonge le « tronc » dans lequel se trouve b2 et lui prend le marquage d'objet le plus élaboré.⁴⁸

Avant modification, l'utilisateur voit la ressource R4 comme quand l'utilisateur visite R1 juste après que « R1 refuse R3 et d2 », sauf que, avantageusement, il n'y a ni b3, ni e3, ni d2, puisque R4 ne les a pas comme ancêtres⁴⁹. Ceci est présenté à la figure 34.

30 Après modifications, l'utilisateur voit la ressource R4 comme le présente la figure 35. a2 est modifié en **a4** et b2 en **b4**. Les objets alternatifs sont b2, e3 (l'absence de c) et d2. A noter que b3 n'est pas présenté car il n'est ni ancêtre ni descendant de b4.⁵⁰

L'utilisateur voit maintenant la ressource R1 comme le présente la figure 36, et la ressource R2 comme le présente la figure 37.

35 L'utilisateur peut préférer un objet alternatif à un objet de remplacement qui lui est présenté. Supposons que l'utilisateur accède à la ressource R1 après que « R1 refuse R3 et d2 » puis sélectionne l'objet alternatif b3 et veuille le modifier en b4. Par la Règle 0 une ressource dérivée R4 est créée et on obtient la structure de la figure 38 dans laquelle la branche dans laquelle se trouve b3 est prolongée par b4 (qui prend de b3 le marquage d'objet le plus élaboré).

⁴⁸ Puisque c'est b2 (objet de remplacement présenté d'office en lieu et place de b1) qui a été modifié en b4, c'est bien le « tronc » (contenant b2) qui est prolongé et non pas la branche qui a été créée pour l'objet refusé b3.

⁴⁹ Ce ne sont donc pas des objets alternatifs prioritaires et par défaut il ne sont alors pas présentés. Mais l'utilisateur peut les faire apparaître en tant qu'objets alternatifs non prioritaires.

⁵⁰ En variante, la mise en œuvre du serveur collaboratif peut aussi présenter b3 en tant qu'objet alternatif moins prioritaire, en considérant qu'il est descendant de b1 (car la ressource R4 dérive de R1).

Après sélection de l'objet alternatif b3 n⁵¹ modification (donc R4 n'est pas encore nécessairement créé), l'utilisateur voit la ressource R1 comme le présente la figure 39.⁵¹ Après modification (ce qui engendre la ressource R4), l'utilisateur voit la ressource R4 comme le présente la figure 40, R1 comme le présente la figure 41 et enfin R2 comme le présente la figure 42.

Supposons enfin que les objets alternatifs présentés dans R1, à savoir : b4, e3 et d2, soient acceptés dans R1. On a alors la structure de la figure 43.

R1, R2 et R4 présentent les mêmes objets à l'utilisateur, à savoir : r2, a2, b2, c2 et d2, et enfin b3 comme objet alternatif. Quant à R3 elle présente r2, a2, b3 (b2 étant alternatif), c2 et d2.

Comme c2 sera logiquement refusé par l'utilisateur puisqu'il vient d'accepter l'absence de c⁵², en guise d'optimisation le serveur collaboratif peut le refuser d'office et on obtient ainsi la figure 44.

Variante à la Règle 3 (fig. 45)

Dans cette variante, la règle suivante 3v remplace la règle 3 énoncée précédemment:

Règle 3v : Dans le cas où la modification d'un objet est acceptée (à l'amont), cette modification est propagée à l'objet acceptant. Si ledit objet acceptant et ledit objet modifié accepté sont dans des branches différentes, alors ledit objet acceptant ainsi que tous ses descendants jusqu'au premier objet à partir duquel un branchement aboutit (directement ou indirectement) à la branche dans laquelle se trouve ledit objet modifié accepté sont déplacés pour se mettre dans la branche où se trouve ledit objet modifié accepté.

Pour l'exemple précédent où le propriétaire de R1 accepte les objets alternatifs b4, e3 et d2, le résultat se trouve maintenant à la figure 45. Au niveau de R1 :

- b1 est bien remplacé par b4, mais ce dernier est déplacé (vers la droite dans la figure) pour se mettre dans la branche où se trouve b3 ; à noter que b4 qui se trouve dans R4 n'a pas été supprimé ; en conséquence, lorsque R1 sera visité par l'utilisateur, c'est l'objet de remplacement b4 (et non pas b2) qui lui sera présenté ;
- e3 se déplace (vers la droite dans la figure) pour se placer dans la branche où se trouve e3 de R3 ; à noter que e3 de R3 n'a pas été supprimé ; en conséquence, lorsque R1 sera visité par l'utilisateur, c'est « l'objet de remplacement » e3 (et non pas c2) qui lui sera présenté.

Remplacement/Enrichissement de Grains (fig. 46 à 49)

On va maintenant prendre en compte le fait que

- les objets sont potentiellement composites,
- qu'il y a la cohérence entre les différentes parties d'un objet composite à vérifier avant de décider si une propagation doit se faire ou pas,
- et qu'en ce qui concerne la discussion par rapport à la Règle 3, il vaut mieux prendre selon les cas, soit l'approche de mise en œuvre présentée en premier (Règle 3), soit l'approche de sa variante (la Règle 3v présentée ci-dessus).

C'est l'utilisateur qui définit la « granularité » des objets composites manipulés.

⁵¹ Variante de mise en œuvre : Après sélection de l'objet alternatif b3 mais avant sa modification R4 est directement créé, l'utilisateur voit alors la ressource R4.

⁵² En revanche, pour b2 on ne peut pas se prononcer aussi facilement: le propriétaire de R1 peut décider de le refuser ou pas. Toutefois, dans la mesure où il s'agit d'objets atomiques (par opposition à des objets composites – ceci sera décrit plus loin), compte tenu du fait que R1 avait préféré b4 à b2, on peut effectuer la même optimisation que pour c, c'est-à-dire refuser b2 d'office.

Au lieu d'objets unitaires comme décrit⁵³ 35 objets de remplacement et les objets alternatifs seront maintenant des grains⁵³. On ne marquera comme étant le plus élaboré (le cas échéant) non seulement des objets unitaires mais aussi des grains⁵⁴.

Définition : Un grain est un *objet* composite dont les modifications (des objets qu'il contient) sont propagées⁵⁵ en « tout ou rien » ; c'est-à-dire qu'elles sont

- 5 • (cas 1) soit entièrement propagées vers un objet correspondant à l'aval – cas où il n'y a pas d'incompatibilité de propagation ;
- (cas 2) soit elles ne sont pas propagées du tout vers un objet correspondant à l'aval – cas où il y a incompatibilité de propagation.
- 10 Il y a incompatibilité de propagation lorsque les modifications propagées, à partir du grain source vers des objets avals, sont différentes pour au moins un objet aval
 - déjà modifié localement
 - ou auquel une modification a déjà été propagée à partir d'une source plus proche (ceci est défini plus loin) que celle dudit grain source.

- 15 A noter que dans l'interface utilisateur, plutôt que (ou en plus) de lui permettre de spécifier quels objets sont (explicitement) des grains et lesquels ne le sont pas, lorsque l'utilisateur demande au système de réifier un contenu contenant des parties elles-mêmes réifiées, on lui permet de dé-réifier les parties réifiées contenus dans la sélection, ou sinon de préciser s'il veut que les changements des parties réifiées soient nécessairement propagées ensembles
- 20 (lorsque l'aval n'est pas incompatible). S'il précise que les changements doivent se propager ensembles alors cela veut dire que l'objet en question est un grain.

Pour décrire le traitement des grains, on va considérer l'exemple d'un objet grain « a »⁵⁶ contenant deux objets : « b » et « c » (figure 46).

- 25 La figure 47 présente un exemple dans lequel un tel grain **a1** existe dans une ressource R1, puis est dérivé en **a2** dans une ressource R2 (dérivée de R1), puis encore dérivée en **a3** dans R3.

Dans cet exemple, tout d'abord, dans R3, c1 est modifié en c3 – ceci est montré dans la première partie de la figure 47 (en haut de la figure).

La deuxième partie de la figure 47 présente deux cas pour la suite de l'exemple:

- 30 Cas 1 : seul b1 est modifié dans R2 (pour donner b2).
- Cas 2 : b1 et c1 sont tous deux modifiés dans R2 (pour donner b2 et c2).

Voici comment ces modifications se propagent vers l'aval :

- 35 • Dans le premier cas, la modification de l'objet b du grain **a2** est propagée vers le grain **a3**. Ainsi, on voit dans le grain **a3** le résultat de la fusion des modifications des objets des grains **a2** et **a3** : le grain **a3** contient les valeurs a1, b2 et c3.
- Dans le deuxième cas, la modification de l'objet b (b1 est modifié en b2) du grain **a2** ne peut pas être propagée dans le grain **a3**⁵⁷. En effet, par définition du concept de grain, b2 et c2 vont de pair, ce qui signifie que le fait de remplacer, dans le grain **a3**, b1

⁵³ Dans la suite, par abus de langage, par « grain » on entendra ensemble d'objets spécifiés comme formant un grain ou objet ne faisant partie d'aucun ensemble d'objets spécifiés comme formant un grain. Ainsi, lorsque l'on parlera de grains, on inclura les objets qui ne font pas partie d'un grain explicitement spécifié comme tel.

⁵⁴ Lorsqu'un objet est dans un grain, seul ledit grain est candidat à être marqué comme étant le plus élaboré. Dans le cas où un objet unitaire n'est pas dans un grain, il peut être marqué lui-même.

⁵⁵ (par le mécanisme de transclusion présenté plus loin)

⁵⁶ Les grains sont notés en gras.

⁵⁷ En fait, par sa conception même, le mécanisme de transclusion (présenté plus loin) n'effectuera pas cette propagation. Ceci est décrit à la section présentant la transclusion en « push ».

par b2 sans aussi remplacer c3 par : t de fournir un résultat incohérent⁵⁸, or comme c3 résulte d'une modification dont la source est plus proche (en fait ici la source est l'objet lui-même, puisqu'il s'agit d'un changement local) il n'est pas question de le remplacer par c2. Ainsi dans le grain **a3**, on maintient les valeurs a1, b1 et c3.^{59, 60}

Les règles Règle 1 à Règle 6 énoncées dans les sections précédentes s'appliquent maintenant aux grains plutôt qu'aux objets qu'ils contiennent. On va remplacer le mot *objet* par le mot *grain* dans ces règles et remplacer la règle 3 (ou sa variante 3v) par la suivante :

Règle 3g : Dans le cas où la modification d'un grain est acceptée (à l'amont), cette modification est propagée au grain acceptant. Ensuite,

- chaque grain à l'aval du grain accepté peut être supprimé (et devenir ainsi un grain implicite) dans l'arbre de modification des grains, s'il reflète⁶¹ déjà le contenu du grain qui se trouve immédiatement à l'amont par rapport à lui-même⁶². Dans le cas où un grain ainsi supprimé était (marqué comme étant) *le plus élaboré*, le grain le plus proche à l'amont (dans la même branche) le devient à sa place.
- Si ledit objet modifié accepté n'est pas ainsi supprimé et s'il se trouve dans une branche différente de celui dudit objet acceptant⁶³, ce dernier ainsi que tous ses descendants (non supprimés) jusqu'au premier objet à partir duquel un branchement aboutit (directement ou indirectement) à la branche dans laquelle se trouve ledit objet modifié accepté sont déplacés pour se mettre dans la même branche que ledit objet modifié accepté.

On va par ailleurs ajouter une règle supplémentaire :

Règle 7 : Lorsqu'une modification d'un objet dans un grain rend le grain correspondant à l'aval incompatible, ce dernier est refusé d'office.⁶⁴ (Cette règle sera illustrée plus loin à la section « Algorithmes en Push »).

La figure 48 montre que, dans le cas où le grain **a3** est refusé par R1 ou R2, à partir de **a2**, une nouvelle branche se crée à partir de la branche où se trouvent les grains **a1** et **a2**, pour contenir **a3**.

Comme le montre la figure 49, dans le cas où le grain **a3** est accepté par R1 (en application de la Règle 3g) :

- Dans le premier cas, les grains a de R2 et R3 peuvent être supprimés puisqu'ils héritent du grain a de R1 composé des objets a1, b2 et c3.⁶⁵

⁵⁸ Un exemple concret en est illustré juste un peu plus loin.

⁵⁹ En effet, il y a toujours b1 car la modification en b2 n'a donc pas pu être prise en compte.

⁶⁰ A noter cependant que si ensuite l'objet a de R1 est modifié en **a2**, cette modification sera propagée au grain **a2** de R2 et à partir de là au grain **a3** de R3, ceci bien que le couple de valeurs b2-c2 de **a2** (dans R2) soit incompatible avec la valeur c3 de **a3** (dans R3). En effet, on peut imaginer que a2 ne joue ici que le rôle d'intermédiaire pour faire transiter une modification qui vient plus de l'amont et qu'il ne serait pas avantageux qu'il « bloque » la propagation de l'amont.

⁶¹ (automatiquement, par transclusion)

⁶² On est alors dans le cas 1 de l'exemple présenté ci-dessus.

⁶³ (en conséquence du fait qu'un des objets qui sont à l'amont dudit objet modifié accepté et à l'aval dudit objet acceptant a refusé ledit objet modifié accepté)

⁶⁴ Cette règle 7 unifie les notions d'« objet le plus récemment modifié » et d'« objet le plus élaboré ». En effet, toute modification qui ne se propage pas aux objets qui sont à l'aval (car étant incompatible) refuse d'office ces derniers, et la modification en question s'effectue donc ainsi sur l'objet le plus à l'aval de la branche active. L'objet modifié est donc forcément tant l'objet « le plus élaboré » que l'objet « le plus récemment modifié » de la branche active (puisque tous les autres qui sont plus à l'aval sont chassés dans une autre branche !).

⁶⁵ (grâce au fait qu'il n'y a pas d'incompatibilité entre

- les objets b2 et c3 qui font partie du grain a de R1
- et ni la modification qui a résulté en l'objet b2 dans R2, ni la modification qui a résulté en l'objet c3 dans R3)

- En revanche, dans le deuxième cas
 - comme le contenu du nouveau grain **a** de R1 (composé des objets a1, b1 et c3) ne peut pas être propagé vers le grain **a** de R2 (composé de a1, b2 et c2) à cause de ses modifications locales (b2 et c2), le grain **a** de R2 ne reflète pas le nouveau grain **a** de R1 et ne peut donc pas être supprimé
 - et comme le grain **a** de R3 est dans une branche différente (dû au fait qu'il avait été refusé), ledit nouveau grain **a** de R1 est déplacé⁶⁶ dans la branche où se trouve le grain **a** de R3 qu'il a accepté ;
 - par ailleurs, le grain **a** de R3 (a1 b1 c3) ne peut pas être supprimé non plus puisqu'il ne reflète pas le contenu du grain **a** de R2 (a1 b2 c2).

La section suivante illustre ce procédé par un exemple concret.

Créneaux (fig. 50)

On peut adopter une structure de « créneaux » dans les valeurs (« innertext »⁶⁷) des objets parents, chaque créneau étant un repère pour y inclure le contenu de l'objet enfant qui lui correspond. L'avantage essentiel de cette approche est qu'elle permet de considérer tout *déplacement* ou *suppression* d'un objet enfant comme une simple *modification* concernant le(s) créneau(x) correspondant(s) de son (ou ses) objet(s) parent(s)⁶⁸.

La figure 50 présente ainsi une ressource R contenant un objet (a) qui lui-même contient deux objets (b et c) exemple dans lequel, les innertexts respectifs sont comme suit :

- a : « J'aimerais lire un beau livre sur {b} {c} »
- b : « les premières voitures »
- c : « à traction avant »,

ce qui permet de présenter dans R le texte « J'aimerais lire un beau livre sur les premières voitures à traction avant ».

On va maintenant illustrer à nouveau l'avantage de spécifier que a (avec ses objets enfants b et c) forme un grain. Reconstruisons les deux cas de l'exemple de la section précédente.

Appelons R1 la version initiale de R (avec les objets a1, b1 et c1 correspondant à a, b et c).

R1 est dérivé en R2, puis R2 est dérivé en R3, et puis dans R3, « à traction avant » (c1) est modifié en « à propulsion » (c3), ce qui résulte en la phrase « J'aimerais lire un beau livre sur les premières voitures à propulsion ».

Ensuite dans R2,

- dans le premier cas, seul « les premières voitures » (b1) est modifié en « les vieilles automobiles » (b2), ce qui résulte en « J'aimerais lire un beau livre sur les vieilles automobiles à traction avant » (a1 b2 c1)
- dans le deuxième cas, « les premières voitures » (b1) est modifié en « les chapeaux » (b2), et « à traction avant » (c1) est modifié en « melon » (c2), ce qui résulte en « J'aimerais lire un beau livre sur les chapeaux melons » (a1 b2 c2).

Alors, dans R3,

⁶⁶(en application du deuxième point de la Règle 3g)

⁶⁷(au sens notamment d'XML)

⁶⁸(le parent avant déplacement peut être différent du parent après déplacement – c'est le cas du déplacement d'un objet enfant d'un premier parent pour le placer au sein d'un autre objet qui sera le nouveau parent dudit objet enfant)

- dans le premier cas on a « J'aime le livre sur les vieilles automobiles à propulsion » (a1 b2 c3)⁶⁹

5 dans le deuxième cas, dû au fait que a est un grain, la propagation ne peut se faire se faire et on a toujours « J'aimerais lire un beau livre sur les premières voitures à propulsion » (a1 b1 c3). En effet, le contenu du grain de R2 n'a pas pu être propagé vers R3, vu que :

- « à propulsion » ne peut être remplacé puisqu'il résulte d'une modification locale
- un grain solidarise a1, b2 et c3, ce qui signifie qu'on ne peut propager b2 sans modifier c3 en c2 – en effet, on voit bien que combiner b2 (« les chapeaux ») avec c3 (« à propulsion »), pour former « J'aimerais lire un beau livre sur les chapeaux à propulsion », serait incohérent – or la source de c3 est plus proche (en fait la source est l'objet lui-même puisqu'il s'agit d'un changement local).

10 Lorsqu'un objet possède un « innertext » morcelé car entrecoupé d'objets enfants, l'approche des créneaux est avantageuse car elle considère l'innertext comme étant globalement atomique. En revanche, quand ce n'est pas le cas (c'est-à-dire quand il n'y a pas de parties d'innertext à gérer), l'approche qui consiste à considérer les déplacements ou suppressions d'objets comme des modifications d'eux-mêmes et non pas de leur parent est bien plus avantageuse car elle permet d'opérer à une granularité plus fine (l'avantage est que l'objet parent n'est pas globalement « gelé » lorsqu'un objet enfant dudit parent est déplacé ou supprimé).

20 Donc selon le cas, le procédé mettra en oeuvre les créneaux ou pas. L'activation de l'option créneau (respectivement non-créneau) peut être automatique en fonction du cas comme expliqué ci-dessus.

25 Dans la suite on utilisera les termes « objet » et « grain » de manière interchangeable sauf quand la distinction entre eux est explicitée.

Procédé de tri d'objets alternatifs par notation (rating)

30 Lorsque l'utilisateur sélectionne un *objet alternatif* plutôt qu'un autre objet qui lui est présenté⁷⁰, la notation du propriétaire de la ressource contenant ledit objet sélectionné est incrémentée⁷¹ et les notations des propriétaires des ressources contenant les objets non sélectionnés sont décrémentés.

Pour mettre ceci en oeuvre, la notation (rating) du propriétaire de la ressource contenant chaque objet sélectionné par l'utilisateur est maintenu dans l'environnement informatique⁷². Ces notations sont relatives audit utilisateur⁷³.

Ainsi les objets alternatifs peuvent être davantage prioritisés automatiquement.⁷⁴

35 Pour des raisons de cohérence, il est préférable que quand un objet alternatif est présenté, les objets alternatifs provenant du même propriétaire soient choisis et que ceux-ci soient même

⁶⁹ Car puisque « à propulsion » peut remplacer « à traction avant » (dans le contexte de « les premières voitures »), à priori il doit aussi aller avec « les vieilles automobiles » qui selon l'utilisateur va même encore mieux avec « à traction avant ».

⁷⁰ (i.e. plutôt que l'objet initialement contenu dans la ressource, ou que l'objet de remplacement ou un autre objet alternatif présenté)

⁷¹ De même, lorsque l'utilisateur dérive une ressource dans laquelle se trouve un objet de remplacement (l'objet de remplacement persiste par défaut dans la page dérivée) et que l'utilisateur ne le remplace pas, le propriétaire de la ressource d'où provient cet objet de remplacement voit sa notation incrémentée.

⁷² Ces notations seront par exemple mémorisées et maintenues dans un serveur de notation.

⁷³ Une notation agrégée pour un groupe d'utilisateurs est aussi envisageable.

⁷⁴ Ils peuvent être triés lors de leurs présentation à l'utilisateur et celui qui est le mieux noté peut être sélectionné pour être présenté par exemple tout seul.

choisis parmi les objets alternatifs corr
même ressource présentée à l'utilisateur.

autres objets qui se trouvent dans la

Procédé de tri d'objets alternatifs par pertinence contextuelle

5 L'histoire des accès à des objets (ou grains)⁷⁵ par l'utilisateur (que l'on appelle navigation) peut être modélisée à l'aide d'une « pile de contexte », où en théorie à chaque objet (pouvant être présenté à l'utilisateur) est associé un score de pertinence (quelle que soit la méthode utilisée pour la calculer) pour l'utilisateur à chaque niveau de navigation, et quand un objet est inexistant dans la pile de contexte on l'assimile à un objet dont le score est égal à zéro.

10 Quand l'utilisateur navigue pour accéder à un nouvel objet, le système⁷⁶ ajoute un niveau à la pile de contexte.⁷⁷

15 Pour un objet donné, le score contextuel est une moyenne des scores non contextuels⁷⁸ à chaque niveau de la pile de contexte, pondérés en fonction de la profondeur. Afin de ne pas avoir à recalculer tous les scores à chaque fois, on utilise une pondération exponentielle, ce qui fait que le score contextuel à un certain niveau est la moyenne pondérée du score non contextuel à ce niveau et du score contextuel au niveau précédent.

Autrement dit, pour un objet donné, s étant le score non contextuel au dernier niveau et r le score contextuel au niveau précédent, le score contextuel au dernier niveau est : $\lambda.r + (1 - \lambda).s$ (λ étant une pondération constante entre 0 et 1, en principe inférieure à $\frac{1}{2}$: plus λ est grand, plus le passé a d'importance).

20 Dans la mesure où l'on connaît pour chaque objet alternatif le score non contextuel de chaque autre objet connu dans le serveur collaboratif par rapport à lui, on peut sélectionner l'objet alternatif pour lequel les scores non-contextuels des autres objets par rapport à lui sont les plus proches du contexte de navigation (c'est-à-dire les plus proches des scores contextuels au dernier niveau de la pile de contexte).

25 A noter qu'au lieu de considérer tous les autres objets on peut se satisfaire d'un ensemble (plus restreint) d'objets représentatifs, par exemple des catégories, et sélectionner l'objet alternatif le plus proche des objets représentatifs du contexte.⁷⁹

Procédé de remplacement par co-identification

Les ressources qui constituent un ERC peuvent être

- 30
- partagées par un ensemble d'utilisateurs (au moins en lecture seule, par exemple ces ressources peuvent être les pages Web constituant un intranet d'entreprise),
 - ou ne pas être partagées (par exemple les fichiers et répertoires non partagés d'un ordinateur individuel) et ne sont alors pas accessibles par tous les utilisateurs de l'ERC.

35 Une ressource non partagée doit être importée, par exemple reçue par courrier électronique, pour bénéficier de la fonctionnalité de remplacement automatique. Dans ce cas, s'il contient un objet plus récent que –et co-identifié⁸⁰ avec– un objet existant, alors il remplace ce dernier. En effet, l'objet importé co-identifié est placé dans l'arbre de modification des objets et devient le plus élaboré au sens de la Règle 5.

⁷⁵ En variante, ce procédé s'appliquera à la granularité des ressources plutôt qu'à la granularité des grains.

⁷⁶ (de préférence il s'agit ici de l'outil de manipulation d'information)

⁷⁷ En revanche, quand par exemple il clique sur la commande « Back » de son navigateur Web le système dépile un niveau.

⁷⁸ (c'est-à-dire déterminés en ne tenant pas compte du contexte)

⁷⁹ Par sélectionner on entend aussi trier selon ledit critère de sélection.

⁸⁰ Par objets co-identifiés on inclut les objets ayant une même description discriminante, au sens des articles de Guha, obtenue par négociation sémantique.

Un objet nouvellement créé joue le même rôle que l'objet importé, à l'égard de ses objets co-identifiés.

Pour être co-identifiable, un objet doit au moins être caractérisé, en d'autres termes : les méta-données qui le caractérisent doivent être renseignées. On propose ici un procédé original analogue à la « brosse de style » (pour reproduire la mise en forme) d'un outil comme Microsoft Word, pour propager des informations standard de description d'objet entre différents objets, caractérisé en ce qu'à chaque objet sont associées des méta-données de description, et en ce que le procédé comprend les étapes suivantes :

- l'entrée dans un mode de propagation à l'initiative de l'utilisateur,
- la sélection d'un objet dont on souhaite emprunter les méta-données de description,
- la sélection d'au moins un autre objet pour lui affecter les mêmes méta-données de description.

L'interface utilisateur est adaptée pour masquer à l'utilisateur les méta-données de description.

Le même procédé permet de propager des méta-données de contrainte de publication, comme décrit plus loin à la section « Méta-données de contrainte de diffusion ».

Procédé de transclusion

Dans la mise en œuvre que l'on décrit ici, le contenu de chaque ressource dérivée selon la Règle 0 (voir plus haut la section « Création de ressource dérivée ») est transclus à partir de ladite ressource accédée en vue d'être modifié.

Rappelons tout d'abord que la transclusion, au sens courant du terme, est l'inclusion par référence. Un exemple classique en est l'application de la balise « img » du langage HTML qui permet d'inclure une image par référence. La transclusion consiste à créer une vue (« le résultat ») d'au moins un objet (« source ») et de placer cette vue dans un autre objet (« le contenant »).^{81, 82}

La technologie de transclusion que l'on décrit ici permet de modifier le résultat de la transclusion. En effet, l'utilisateur doit pouvoir modifier le contenu d'une page dérivée (comme décrit jusqu'ici).

L'utilisateur peut définir les modifications à effectuer en agissant directement sur le résultat de la transclusion, en utilisant un outil commun de manipulation de contenu, par exemple l'outil qui lui permet de manipuler le contenant (et son contenu local) ou encore l'outil qu'il utiliserait s'il voulait modifier la source. Lorsque l'utilisateur modifie ainsi directement le résultat d'une transclusion, les résultats des actions qu'il effectue (tel élément est supprimé, le contenu de tel autre est modifié, un autre encore est déplacé et modifié...) sont enregistrés afin de pouvoir retrouver ces modifications automatiquement la prochaine fois que l'objet est mis à jour. Par exemple si l'utilisateur échange deux parties d'un objet transclus, lorsque le contenu de cet objet se modifie à la source, les deux parties vont conserver leur nouvelle position mais leur contenu va toujours correspondre à celui de la source.

⁸¹ Par commodité, il arrive dans la suite que l'on écrive « transclusion » au lieu de « résultat de transclusion ». D'autre part, par « objet transclus » on entend « objet résultant de transclusion » (par opposition à objet source) sauf quand le contexte fait comprendre le contraire.

⁸² Les objets sources peuvent être distants. Ils sont soit identifiés directement, soit implicitement en tant que résultat d'une requête de recherche. Par exemple, on peut inclure dans un document Word le résultat d'une requête à une base de données. Dans la suite par « la source (d'une transclusion) » on entend « l'objet source ou l'ensemble des objets sources (de ladite transclusion) ». Lorsque le contenant est visionné par la suite, une présentation de la source sera visible à l'emplacement de la transclusion. Autrement dit, chaque fois que l'objet contenant sera ouvert ou actualisé manuellement ou automatiquement, la version courante (c'est-à-dire la plus récente) de la source sera présentée à cet emplacement.

Pouvoir modifier le résultat de la transclusion⁸³ de la même manière que si l'on modifiait une copie locale de la source⁸³, mais sans perte du lien avec la source, représente la caractéristique primordiale de la technologie de transclusion décrite ici.

- 5 On va maintenant décrire certains concepts qui seront nécessaires pour comprendre la mise en œuvre du procédé, puis décrire les algorithmes qui permettront d'actualiser chaque transclusion de la version courante (c'est-à-dire de mettre à jour la ressource dérivée en question).

Identifiant

- 10 Chaque objet possède un identifiant. Les objets créés localement possèdent un identifiant *atomique*, qui est un nombre entier unique dans le système.

De même, chaque transclusion possède un identifiant atomique, unique également.

Les objets vus à travers une transclusion possèdent un identifiant *composite*, noté A:B, où A (la tête de cet identifiant) est l'identifiant de la transclusion, et B (la queue de cet identifiant) est l'identifiant de l'objet à la source de cette transclusion.

- 15 On utilise des parenthèses pour clairement délimiter la tête et la queue en cas de besoin. En l'absence de parenthèses, les sous-identifiants s'étendent le plus à droite possible. Par exemple, 1:2:3:4 doit être lu 1:(2:(3:4)). Autre exemple, si A=1:2 et B=3:4 alors A:B=(1:2):3:4.

Transclusion

- 20 Dans la présente section présentant les algorithmes, une transclusion A est considérée comme d'une part un ensemble d'objets « vu à travers » A, et d'autre part à un ensemble de modifications (Differences) concernant ces objets.

La spécification d'une transclusion est constituée des deux parties suivantes :

- 25 1. le résultat stocké, dans la suite appelée « version courante »⁸⁴, de la transclusion construit d'après la dernière version accédée de la source ;
2. la spécification des modifications (dans la suite appelée « différences »⁸⁵) appliquées au résultat de la transclusion.

- 30 La *version courante* du résultat d'une transclusion est le résultat de la transclusion tel qu'il a été présenté la dernière fois que la page a été affichée. Cette version courante est conservée, et est typiquement mise à jour par le système à chaque fois que l'utilisateur y accède.

Il peut être intéressant de ne stocker dans la version courante que les parties créées localement, et pour le reste de ne garder que les références d'où il peut être obtenu.

- 35 Les *différences*, quant à elles, indiquent la ou les sources de cette transclusion, ainsi que les modifications à y appliquer. Dans l'approche décrite ici, les différences s'appuient sur la version courante⁸⁶ et n'ont ainsi pas besoin de contenir « in extenso » les contenus et les positions des objets à modifier.

Typiquement, une fois que les modifications apportées par l'utilisateur (dans la page dérivée) au contenu du résultat d'une transclusion ont été détectées et enregistrées dans les différences,

⁸³ (le résultat restant quand même tenu à jour par rapport à la source, mais les modifications faites sur le résultat étant prioritaires par rapport aux modifications apparues à la source)

⁸⁴ (ou encore « contenu de la ressource dérivée »)

⁸⁵ (ou encore « changements »)

⁸⁶ Les différences se réfèrent aux objets de la version courante (c'est-à-dire à leurs identifiants respectifs) pour définir les transformations qui leurs sont appliquées.

la prochaine fois que ladite page dé⁴²idée (ou simplement rafraîchie), ces modifications sont re-appliquées à une version mise à jour de la source, tout en se servant du contenu de ladite page dérivée (version courante) pour avoir les contenus et positions des objets modifiés.

5 Ces modifications incluent

- modification ou suppression d'un objet vu à travers A (donc dont l'identifiant commence par « A: »)
- déplacement d'un objet vu à travers A, dont le parent avant et après déplacement est également un objet vu à travers A.
- 10 • création d'un objet (créé localement ou vu à travers une transclusion différente de A) comme enfant d'un objet vu à travers A.
- déplacement d'un objet vu à travers A comme enfant d'un objet créé ou vu à travers une transclusion différente de A. Un cas particulier est le suivant : lorsque l'utilisateur effectue une transclusion, l'algorithme, pour effectuer cette opération, va d'abord créer la transclusion mais à cet instant cette dernière n'est pas encore accessible (car aucun objet vu à travers cette transclusion n'est encore placé comme enfant d'un objet accessible.) La seconde étape n'est alors rien d'autre qu'un déplacement d'un de ces objets inaccessibles comme enfant d'un objet accessible.
- 15

20 A noter qu'en cas de déplacement d'un objet d'une transclusion à une autre ou de création d'une transclusion dans une autre transclusion, cette modification sera « partagée » par deux transclusions.

A noter également que lorsqu'un descendant de résultat de transclusion est déplacé hors de ce résultat, il faut créer (en plus de l'information de déplacement dans les différences de la transclusion) un second objet « résultat de transclusion ». Dans la terminologie XML décrite 25 ci-dessous, cela signifie qu'il y aura deux nœuds « Transclude » avec le même identifiant de transclusion (attribut « id »).

Par convention, on dit qu'une transclusion implicite ne contient aucune modification. En effet, on dira que de telles modifications seront traitées par la transclusion par laquelle cette transclusion implicite est venue. Par exemple si un objet ((1:2):3):4 est modifié, cette 30 modification sera associée à la transclusion 1.

Les différences peuvent être en format XML comme illustrée par l'exemple suivant :

```

<Differences id="8">
  <FirstSentences id="8" count="3" keepChildren="true" />
  <Modify id="8:2" />
  35  <Move id="8:3" />
  <Modify id="8:6:4:3" />
  <Move id="8:6:4:3" />
  <Create id="9" />
  <Delete id="8:5" />
  40  <Transclude id="10" ref="document7#7" />
</Differences>

```

Dans cet exemple, le nœud Differences indique l'identifiant de la transclusion, qui est ensuite référencé dans un nœud Transclude, à l'emplacement du résultat.

45 Ensuite tous les nœuds enfants de ce nœud Differences indiquent les changements à appliquer sur la source de la transclusion pour construire le résultat. Les identifiants « id » se réfèrent à la version courante de la transclusion.

Dans l'exemple ci-avant, FirstSentences ir qui sélectionne les premières phrases du contenu d'un objet (voir plus loin la section « Transclusions avec transformations »).

Dans le cas d'objets créés (Create), modifiés (Modify) ou déplacés⁸⁷ (Move), le nouveau contenu et/ou position sont obtenus dans la version courante et ainsi ne sont pas spécifiés dans les différences.

Transclude indique une nouvelle transclusion au sein de la transclusion en question et se réfère donc à un autre objet des différences (l'attribut id="10" étant l'identifiant de la transclusion, auquel est associé un autre objet Differences, et ref la source de la transclusion. Dans cet exemple, l'identifiant du résultat de cette transclusion sera donc 10:7. A noter que la source peut être identifiée par une référence ou une requête).⁸⁸

Cet exemple montre aussi que pour déplacer et modifier un même objet (8:6:4:3) on place deux opérateurs dans l'objet des différences.

Tous ces opérateurs peuvent être implantés dans des modules de manière à ce que de nouveaux types d'opérateurs puissent être ajoutés au système afin d'étendre ses possibilités. Dans cet exemple, FirstSentences a pu être sélectionné par l'utilisateur dans une liste (extensible) d'opérateurs au moment de la transclusion ou par la suite en l'y associant.

Rappelons que les changements Move, Modify et Create ont été créées par l'utilisateur en agissant directement sur les objets, de même manière que s'il modifiait une version locale.

L'utilisateur peut demander que le contenu de la source d'une transclusion ne soit pas du tout inclus, mais qu'un simple lien hypertexte soit placé comme résultat, pointant vers la ou les sources. L'objet Differences, dans ce cas, ressemblerait à ceci :

```
<Differences id="18" >
  <HyperLink text="click here" />
</Differences>
```

Cet objet Differences pourra en général être omis puisque la présentation du résultat ne dépend plus du contenu de la source et ne doit plus être mis à jour. En fait on ne traite pas différemment un lien hypertexte placé directement et une transclusion présentée sous forme de lien.

De même, une référence ou une relation (unilatérale) quelconque d'un objet à un autre est une transclusion dont le résultat n'est ni construit ni présenté (ni même ne contient un simple lien hypertexte).

Dans le cas où les informations sont en notation XML, un objet est un élément XML et le contenu de l'objet est l'« innertext » de l'élément XML. Les attributs et les enfants sont tous deux considérés comme objets enfants, toutefois les attributs de même nom d'attribut provenant de sources différentes doivent être fusionnés.

⁸⁷ Dans l'approche des « créniaux » décrite plus haut, il n'y a pas d'opération de déplacement d'objet en tant que tel puisqu'un déplacement d'objet enfant revient à modifier (l'innertext de) son (ou ses) objet(s) parent(s).

⁸⁸ On a vu que le contenu global de chaque ressource dérivée est transclus du contenu global de la ressource source. Mais le procédé de transclusion peut aussi être utilisé pour transclure, dans une page dérivée, un objet quelconque se trouvant dans une page quelconque. C'est ainsi qu'une page dérivée peut servir à assembler des objets situés dans des pages externes. Ainsi, les « différences » associées au résultat de transclusion constituant le contenu d'une ressource dérivée, peuvent elles-mêmes inclure des transclusions. Ces dernières peuvent alors elles-mêmes bénéficier des remplacements et enrichissements provenant des remplacements et enrichissements de la source. Chaque objet du résultat d'une transclusion manuelle bénéficie des avantages principaux décrits ici (à savoir les remplacements et enrichissements), à condition que sa source se trouve dans une ressource gérée dans le cadre du même ERC. Ledit objet sera ainsi automatiquement remplacé par un objet de remplacement selon la Règle 5 et enrichi d'objets alternatifs selon la Règle 6, appliquées à ladite source. Toutefois, dans le cas où pour un objet donné sont déterminés plusieurs objets de remplacement, les objets supplémentaires obtenus à partir de l'aval des transclusions hors dérivation sont considérés comme objets alternatifs.

Transclusion implicite, transclusi

Lorsque le résultat (3:2 dans la figure 51) d'une transclusion (3) est visible à travers une transclusion (4), on dit que l'image de ce résultat (4:3:2) est le résultat d'une *transclusion implicite* (4:3) de l'image (4:2) de la source (2) de la première transclusion (3), et cette première transclusion (3) est appelée transclusion *interne* à la seconde (4).

Dans cet exemple, l'image (4:2) de la source (2) de la transclusion interne est également visible à travers la transclusion (4), mais ce n'est pas toujours le cas. Dans le cas contraire, la source de la transclusion interne sera simplement invisible. L'exemple de la figure 51 en donne une conséquence concrète. Dans cet exemple, on peut dire que 4:3:2 est une transclusion indirecte de 2, car les modifications appliquées sur 2 seront indirectement transmises à 4:3:2.

Lorsque cette structure est vue à travers la transclusion 5, la transclusion indirecte est maintenue, bien qu'elle transite par un objet invisible (5:3:2). Ainsi, des modifications appliquées à 5:2 seront répercutées sur 5:4:3:2.

15 Equivalence d'identifiants

L'existence d'un objet peut parfois être obtenue par plusieurs chemins, en présence de transclusions implicites. Dans l'exemple de la figure 52, 4:3:2 est à la fois l'objet 3:2 vu à travers la transclusion 4, et l'objet 4:2 vu à travers la transclusion 4:3 (3:2 et 4:2 sont les deux sources de l'objet 4:3:2). En considérant la seconde manière, on obtient l'identifiant (4:3):4:2 (on place des parenthèses afin de clairement identifier la tête de cet identifiant).

En pratique, on va considérer ces deux identifiants comme équivalents, et plus généralement que pour tous A, B, C, (A:B):A:C est équivalent à A:B:C.⁸⁹

Calcul des sources d'un objet

Chaque objet ayant un identifiant composite a un ensemble de sources⁹⁰ représentées par différentes formes (équivalentes) $T_i:S_i$ de son identifiant, où S_i est la source et T_i la transclusion par où il a transité. Par exemple, les sources de 4:3:2 sont représentées par 4:(3:2) et (4:3):(4:2).

Le calcul de cet ensemble s'effectue ainsi:

On débute par une forme $T_0:S_0$ de l'identifiant.

30 Soit $T_i':S_i'$ les paires transclusion-source de S_0 . L'ensemble des sources est représenté par $(T_0:T_i'):(T_0:S_i')$ pour chaque i , en plus de $T_0:S_0$.

Origine d'une information

En calculant une structure on va associer deux types d'*information* que sont sa position et son contenu (hormis ses enfants – le fait qu'un objet A est enfant d'un objet B est mémorisé en tant que position de A). Tant pour la position que pour le contenu on y associe une *origine* de la forme⁹¹ P!O.

⁸⁹ On décrit plus bas le concept de densité qui permettra notamment de déterminer l'équivalence entre identifiants.

⁹⁰ Un identifiant composite formé seulement de deux identifiants atomiques n'aura qu'une seule source. Par ensemble on inclut ici le cas d'un ensemble d'un seul élément.

⁹¹ (Chemin!Origine)

Distance de sources

Un objet pouvant avoir plusieurs sources, il va être nécessaire de déterminer l'importance de ces sources de façon à ce que, lorsque plusieurs d'entre elles sont modifiées, on puisse déterminer de laquelle l'objet doit hériter de l'information.

- 5 Le but est qu'un fournisseur "proche" ait un poids plus fort qu'un fournisseur "distant". Ici "distant" signifie à travers une transclusion plus "haute" (englobante); par exemple, la transclusion 4 est plus haute que la transclusion 4:3.

10 **Règle 8** En présence de plusieurs sources candidates pour le contenu (ou la position) d'un objet, on préfère celle dont la proximité est la plus grande (ou celle dont la distance est la plus petite).

Première méthode

Une première méthode se base sur la notion de « densité » d'identifiant.

On note $d(X)$ la densité d'un identifiant X .

- 15 Une densité est de la forme $\sum S_x d_x$ où S_x est un polynôme en s et d_x est le poids de l'identifiant atomique x . Le polynôme S_x ci-dessus est appelé *densité partielle* de x dans X .

La densité est calculée récursivement comme suit (s est un symbole sans valeur numérique définie):

La densité $d(n)$ d'un atome est d_n .

La densité $d(A:B)$ d'un identifiant composite $A:B$ est $(1-s)d(A) + s d(B)$.

- 20 Comme chacun sait, un algorithme de simplification de polynômes consiste à remplacer tous les produits de sommes par des sommes de produit, puis pour chaque x à regrouper les termes en d_x .

Par exemple:

- 25 $d(4:3:2)$
 $= (1-s) d(4) + s d(3:2)$
 $= d4 - s d(4) + s ((1-s) d(3) + s d(2))$
 $= (1-s) d(4) + (s-s^2) d(3) + s^2 d(2)$

De même, la densité de l'autre forme de cet identifiant est la suivante :

- 30 $d((4:3):4:2)$
 $= (1-s) d(4:3) + s d(4:2)$
 $= (1-s) ((1-s) d(4) + s d(3)) + s ((1-s) d(4) + s d(2))$
 $= (1-s) d(4) + (s-s^2) d(3) + s^2 d(2)$

Calcul de la proximité d'un objet par rapport à un autre

- 35 Pour calculer la proximité d'un objet P relativement à un objet R , on remplace dans $d(P)$ chaque d_x par la densité partielle de x dans R . Ceci donne un polynôme en s .

L'algorithme pour comparer de tels polynômes $A = \sum a_i s^i$ et $B = \sum b_i s^i$ est le suivant⁹²:

- 40 1. Initialiser j à 0.
 2. Si $a_j < b_j$ alors $A < B$
 3. Si $a_j > b_j$ alors $A > B$
 4. Sinon incrémenter j et retour à 2.

⁹² Considérer que s tend vers zéro tout en restant positif.

Par exemple, $2s^4 < s^4 + s^3 < 2s^4 + s^3 < s$.

46

On utilise le terme de proximité car plus la valeur est grande plus l'objet est proche.

Deuxième méthode

- 5 Lors de la transclusion d'un objet englobant des objets descendants dont l'un est transclus à partir de l'autre, il faut définir pour la transclusion entre objets descendants de l'objet englobant transclus une valeur de distance plus courte à fixer entre ces objets.

Ainsi la « longueur » d'une transclusion peut se déterminer comme suit. On donne une longueur de 1 aux transclusions non implicites (donc ayant un seul terme dans leur identifiant), une longueur un tout petit peu plus petite $1 - \delta$ (δ tendant vers 0) aux transclusions implicites
10 ayant deux termes, une encore un peu plus petite $(1 - 2\delta)$ aux transclusions ayant 3 termes, et ainsi de suite. Donc on donne la valeur $1 - (n - 1)\delta$ aux transclusions ayant n termes (Par terme on entend les identifiants atomiques composant la transclusion, par exemple (1 :2) :3 :4 contient quatre termes).

- 15 Pour comparer les distances respectives de deux sources, on somme les distances des transclusions par lesquelles chacune transite, puis on compare d'abord leurs termes sans le δ et s'il est le même on compare en plus leurs termes avec le δ . La source la plus proche est ainsi celle pour laquelle la distance est la plus courte.

- 20 On dira que pour une modification locale, la source se trouve à une distance nulle. Ainsi lorsque l'on compare une modification locale à une modification provenant de l'amont, la même règle s'applique et la modification locale a ainsi la priorité par rapport à n'importe quelle autre source de propagation de modification.

Procédé de transclusion en « pull »

On va maintenant décrire une première méthode (dite en « pull ») pour mettre en œuvre la transclusion. Plus loin une autre section en présentera une variante (dite en « push »).

- 25 On décrira tout d'abord un « algorithme local » qui permet de déterminer le contenu et la position d'un certain objet, puis un « algorithme global » qui permet, en s'appuyant sur l'algorithme local, de construire un objet et tous ses descendants.

Il suffira alors, pour mettre à jour une version courante, d'appeler l'algorithme global sur la racine de cette version courante.

30 Calcul d'un objet – algorithme local

Cet algorithme va calculer tant l'*information* (position + contenu) d'un objet donné que l'origine de cette information (séparément l'origine de la position et l'origine du contenu)⁹³.

Entre crochets est noté l'étape dans l'algorithme.

- 35 On calcule les deux types d'information, c'est-à-dire la position et le contenu d'un objet, indépendamment mais l'algorithme étant similaire on va mettre en œuvre un algorithme générique pour calculer ce qu'on a appelé l'*information* (puis le spécialiser).

- 40 On a d'abord besoin de savoir si l'information est produite par l'objet lui-même. Dans le cas d'un objet créé, ceci est toujours vrai. Dans le cas où l'objet provient d'une transclusion, cette dernière maintient une liste d'objets modifiés (dans ses Différences), ce qui permet donc de savoir si l'information a été produite par l'objet lui-même ou pas. [local1]

⁹³ On peut, en variante, adopter l'approche des slots, précédemment décrite, pour considérer que les déplacements sont des modifications de parents, et éviter ainsi de traiter les déplacements de manière différente des modifications.

Si l'information est produite par l'objet l⁴⁷ origine est !X (où X est l'identifiant de l'objet). [local2]

Si l'information n'est pas produite par l'objet (X) lui-même:

5 Soit $T_i:S_i$ l'ensemble des paires transclusion-source pour cet objet et soit $P_i!O_i$ l'ensemble des origines (chemin!origine) correspondant à ces sources⁹⁴. [local3]

On calcule la proximité de chaque O_i par rapport à X. [local4]

Soit O_j le plus proche. L'origine de l'information pour X est $T_j:P_j!O_j$ [local5] et l'information pour X est obtenue de $T_j:S_j$ [local6].

10 Dans le cas où l'information calculée est le contenu de l'objet, il sera simplement copié de $T_j:S_j$.

Dans le cas où c'est sa position, il ne sera pas copié mais les identifiants (parent F et prédécesseur B de S_j) seront étendus avec T_j . Le parent de X sera $T_j:F$. L'information du parent a la priorité par rapport à l'information du prédécesseur. Ainsi si $T_j:B$ n'est pas un enfant de $T_j:F$, X sera placé (par convention) comme dernier enfant de $T_j:F$. Sinon il sera mis
15 (par convention) juste après $T_j:B$. [local7].

Calcul d'une structure d'objets – algorithme global

Chaque objet doit mémoriser le fait qu'il a déjà été traité par l'algorithme local ou global, respectivement, de manière à ce que chaque algorithme ne traite aucun objet plus d'une fois.

On appelle X l'objet sur lequel s'exécute la passe courante de l'algorithme global.

20 Quand X est un objet localement créé (et qu'il a donc un identifiant atomique), son contenu et la liste de ses enfants sont déjà connus, il a donc juste à exécuter l'algorithme global sur ses enfants. L'origine de l'information (pour sa position et son contenu) est bien sûr juste !X. [global1]

25 Si au contraire X provient d'une transclusion, soit $T_i:S_i$ l'ensemble des sources. Alors exécuter l'algorithme local sur les objets suivants :

- X lui-même (pour obtenir son contenu) [global2]
- Pour chaque i, pour chaque enfant C_{ij} de S_i , $T_i:C_{ij}$. [global3]
- Tous les objets que la transclusion a déplacé ou créé [global4].

Ensuite, l'algorithme global est exécuté sur les enfants de X.

30 Procédé de transclusion en « push »

Le procédé de transclusion en « push »⁹⁵ peut être mis en œuvre au moyen de trois tables, gérées dans le serveur collaboratif et dans lesquelles on stocke les différences⁹⁶ :

⁹⁴ L'origine de l'information pour les sources sera typiquement obtenue par un appel récursif à l'algorithme local, sur chacune des sources. L'algorithme local ne boucle pas car il finira toujours par atteindre, en remontant les transclusions, des objets fournissant leur contenu localement, à la limite en atteignant les objets ne se trouvant pas dans un résultat de transclusion.

⁹⁵ Cette approche peut être utilisée pour mettre à jour des résultats d requêtes persistante à une base de données. Prenons le cas d'une base de données RDF (voir la recommandation du W3C) (contenant donc des triplets {objet ; propriété ; objet}) et d'un langage de requête tel que SPARQL (voir la recommandation du W3C). On peut modéliser l'espace des triplets sous la forme d'une structure d'arbre où les nœuds représentent des triplets « abstraits » (chaque élément d'un triplet abstrait représente les ensembles d'objets ou propriétés possibles) et où les feuilles représentent les données existantes dans la base de données. Chaque requête persistante correspond à un sous ensemble de cet espace, c'est-à-dire à un ensemble de sous-arbres (représentés par un ensemble de triplets abstraits). Lorsqu'un ensemble de triplets concrets c'est-à-dire des

- Table des Transclusions
- Table des Changements Locaux
- Table des Changements (provenant) de l'Amont⁹⁷

5 Cette approche permet de propager les changements des objets des sources des transclusions, sur les résultats de ces transclusions, en suivant ces dernières grâce à la table des transclusions.

L'idée est que toute transclusion est indiquée dans la Table des Transclusions, et pour tout résultat de transclusion, dans le cas où il est directement modifié lui-même, l'information de l'existence de cette modification est indiquée dans la Table des Changements Locaux, tandis que dans le cas où il doit être mis à jour suite à une modification faite à l'amont, l'information de l'existence de cette modification ainsi que l'adresse de l'objet modifié⁹⁸ se trouvent dans la Table des Changements de l'Amont.

15 Pour accéder à une ressource R, normalement l'extension de l'outil de manipulation d'information obtient sa version courante à partir du serveur de contenu qui l'héberge. En parallèle il demande à la Table des Changements de l'Amont s'il y a eu des changements pour cette ressource (et pour quels objets il y a eu des changements), auquel cas cette Table retourne⁹⁹ les adresses où les objets en question se trouvent. Alors, l'extension de l'outil de manipulation d'information obtient ces objets et les assemble, au sein de la version courante reçue pour la ressource en question, afin d'obtenir une nouvelle version courante¹⁰⁰.

20 Lorsqu'un changement (local) est effectué sur un objet O se trouvant dans un résultat de transclusion dans une ressource R, le serveur hébergeant R en est informé, l' (ou les) entrée correspondant audit changement est insérée dans la Table des Changements Locaux, et comme tout changement local a priorité sur les mises à jour éventuelles dues à des changements provenant de l'amont, l' (ou les) entrée correspondante est supprimée dans la Table des Changements de l'Amont. Par ailleurs, dans le cas où dans la Table des Transclusions est indiquée une transclusion prenant comme source l'objet O lui-même ou un de ses ancêtres, l'information concernant le changement de l'objet correspondant résultant de cette transclusion est ajoutée dans la Table des Changements de l'Amont. Cette information contient aussi les identifiants de tous les ancêtres dudit objet correspondant, de manière à ce que si une autre transclusion indiquée dans la Table des Transclusions a comme source ledit objet correspondant ou un des ses ancêtres, une autre information soit ajoutée dans la Table des Changements de l'Amont, et ainsi de suite jusqu'à qu'il n'y ait plus de transclusion correspondante dans la Table des Transclusions. Chaque nouvelle information de changement d'un objet résultant d'une transclusion est ajoutée avec l'information de distance entre l'objet source et l'objet résultat. Ainsi, à chaque fois qu'une information de changement est à ajouter dans la Table des Changements de l'Amont pour un même objet résultat, elle est effectivement

feuilles de cet arbre sont modifiées, on va propager cette modification en push, c'est-à-dire re-exécuter les requêtes persistantes concernées qui sont simplement les ancêtres des triplets concrets modifiés.

⁹⁶ Chaque différence (qu'il s'agisse d'un changement local ou une conséquence d'un changement à l'amont) peut être caractérisé par le type de changement, l'identifiant de l'objet changé et les identifiants de ses ancêtres...

⁹⁷ A noter que grâce à la Règle 8 (qui prévoit une distance nulle pour les modifications locales) la table des changements de l'amont pourrait suffire.

⁹⁸ Si on n'est pas dans l'approche des créneaux, s'il s'agit d'un déplacement ou d'une suppression d'objet on va mémoriser les identifiants du parent et du prédécesseur (éventuel), et l'algorithme va retenir séparément la source ayant le plus court chemin pour le contenu et pour la position.

⁹⁹ Si on n'est pas dans l'approche des créneaux et qu'il s'agit d'un changement de position, le parent et le prédécesseur (éventuel) sont retournés.

¹⁰⁰ Avantagusement cette dernière est ensuite communiquée au serveur qui héberge la ressource R, de manière à ce qu'elle remplace la version courante précédente, et que la Table des Changements de l'Amont puisse indiquer, pour les entrées correspondantes à ces changements, que le changement en question est déjà pris en compte dans la version courante et qu'il ne faut donc pas aller le récupérer. Bien évidemment, lors d'un nouveau changement du même objet amont cette indication est supprimée dans l'entrée en question.

ajoutée seulement si la distance avec la s transclusion considérée est inférieure à la distance de l'information le cas échéant déjà existante (auquel cas elle la remplace).

A noter que dans cette approche, il n'est pas nécessaire de stocker dans la Tables des Changements de l'Amont l'information concernant le changement d'un objet résultant d'une nouvelle transclusion, puisque le changement est alors déjà pris en compte déjà dans la version courante et que quand la source la plus proche sera le cas échéant modifiée à nouveau, une entrée sera alors insérée dans la Table des Changements de l'Amont.

En revanche, dans le cas où les objets sont modifiés fréquemment il peut être avantageux de n'insérer l'information concernant le changement d'un objet résultant d'une transclusion que la première fois, c'est-à-dire quand l'entrée correspondant au changement vient d'être créée dans la Table des Changements Locaux. Dans cette méthode, lorsqu'une nouvelle transclusion est insérée dans la Table des Transclusions, les informations pour tous les objets contenus dans le résultat de la transclusion sont insérées dans la Table des Changements de l'Amont avec leur date de création. Par ailleurs, une date de modification est mémorisée (et maintenue) avec chaque entrée dans la Table des Changements Locaux. A chaque accès à une ressource, la date contenue dans la Table des Changements de l'Amont est comparée avec la date du changement de la source dans la Table des Changements Locaux. Si la modification de la source est plus récente alors cette dernière est demandée à son serveur de contenu et la Table des Changements de l'Amont est mise à jour avec la nouvelle date.

A noter que lors de l'exécution de l'algorithme, il faut non seulement consulter la table des transclusions mais aussi toutes les informations de déplacement dans la table des changements locaux. Les figures 53 et 54 montrent pourquoi c'est nécessaire.

Dans la première des deux schémas de la figure 53, une transclusion de (l'objet) 1 est effectuée, puis 10:4 est déplacé dans le résultat. Ensuite, dans le deuxième schéma des figures 53, 4 est déplacé à la source hors de l'arbre de 1. 10:4 ne disparaît cependant pas du résultat car sa position avait été figée lors du déplacement. Le contenu de 10:4 doit donc rester synchronisé avec celui de 4 (par exemple, si des enfants sont créés pour 4 alors des objets correspondants doivent être visibles comme enfant de 10:4).

Lorsque 4 est modifié, 4 et tous ses ancêtres sont cherchés dans la table des transclusions ainsi que dans l'ensemble des déplacements indiqués dans la table des changements locaux. Ainsi l'entrée enregistrant le déplacement de 10:4 sera trouvée, ce qui permettra de propager cette modification.

La figure 54 présente un deuxième scénario.

Il s'agit ici d'un déplacement de 10:4 hors du résultat de la transclusion. Ici encore on souhaite que la synchronisation entre 4 et 10:4 soit assurée, et cela fonctionne comme dans le scénario précédent si on tient aussi compte du déplacement de 10:4 lors de la propagation de modifications, donc en quelque sorte 10:4 est vu comme le résultat d'une transclusion (virtuelle) de 4.¹⁰¹

Structures mixtes

L'algorithme en push est intéressant lorsque les modifications à l'amont sont moins fréquentes que les accès (ou rafraîchissements) à l'aval, en particulier lorsque la source est à l'intérieur du système et donc que ce dernier¹⁰² peut automatiquement être informé lorsqu'elle est modifiée.

¹⁰¹ Que la version courante soit mise à jour ou non, on peut tenir à jour (dans le serveur de collaboration) la structure entière des résultats de transclusions afin que les listes d'ancêtres soient toujours correctes.

¹⁰² (il s'agit du serveur de collaboration)

Dans certains cas il peut être souhaité . structures mixtes, où des chaînes de transclusions sont partiellement en pull et partiellement en push (chaque transclusion peut fonctionner soit en push ou en pull, indépendamment des autres).

5 Aucun travail particulier n'est nécessaire dans le cas « push-pull », i.e. lors de la transclusion en pull d'un résultat de transclusion qui elle fonctionne en push. En effet, lorsque l'objet à l'amont est modifié, l'objet intermédiaire est notifié (i.e. une entrée est placée dans la table des changements provenant de l'amont). L'objet à l'aval n'est pas notifié, mais, fonctionnant en pull, ce dernier ira consulter l'objet intermédiaire lorsqu'il doit être affiché, et verra le nouveau contenu de ce dernier. Cette configuration peut par exemple être intéressante lorsque
10 l'objet à l'amont est rarement modifié mais l'objet intermédiaire l'est souvent.

La figure 55 montre l'évolution des versions courantes lors d'une modification à l'amont. La première ligne montre l'état avant modification. A la deuxième ligne l'amont est modifié, et la modification est répercutée sur l'objet intermédiaire mais pas l'objet à l'aval. La dernière ligne montre que le changement est observé lors d'une requête à l'aval (le contenu de l'objet intermédiaire étant copié à l'aval).
15

Le cas « pull-push » demande par contre un traitement particulier. En effet, une modification à l'amont ne sera pas propagée à l'objet intermédiaire, par la nature pull de ce dernier, et une requête pour l'objet à l'aval ne provoquera pas de requête à l'objet intermédiaire, la table de changements à l'amont n'ayant pas été modifiée.

20 Cette situation est traitée de la façon suivante :

En construisant la version courante à l'aval on marque les objets dont le contenu (ou la position) a été obtenu par l'amont, avec des références directes vers l'emplacement de ce contenu. Lorsque l'objet à l'aval est affiché, on effectue une mise à jour des parties concernées, envoyant la requête directement à l'objet à l'amont (et si ce dernier est lui-même
25 résultat d'une autre transclusion en pull, la requête pourra continuer depuis ce dernier, en cascade, selon le fonctionnement normal de l'algorithme pull).

Ceci est intéressant si les différences appliquées par l'objet intermédiaire ne changent pas souvent, et qu'au contraire l'objet à l'amont est modifié fréquemment, ou ne peut envoyer les notifications de mise à jour dans les tables de changement (par exemple s'il s'agit d'un objet externe). L'intérêt de cette approche est qu'on a les avantages de la méthode « push » (en évitant les requêtes en cascade pendant que l'utilisateur attend), et les avantages de la méthode pull (que l'objet à l'amont n'est interrogé que lorsque nécessaire).
30

La figure 56 reprend le même scénario que la figure précédente. On voit dès la première ligne que l'objet à l'aval contient une référence à l'objet à l'amont, indiquant que ce contenu devra être rafraîchi (en pull) à chaque requête.
35

Lors de la modification à l'amont, ce dernier n'étant pas référencé par une transclusion en push, aucun contenu n'est transféré. Lors de la requête pour l'objet à l'aval, le contenu est mis à jour, via la référence se trouvant dans la version courante. On voit bien à la dernière ligne que la référence court-circuite l'objet intermédiaire, et que ce dernier n'est jamais mis à jour ou même interrogé, dans ce scénario (en effet, toute modification apportée à l'objet intermédiaire est automatiquement envoyée à l'objet à l'aval, par la méthode push).
40

On peut évidemment avoir des structures arbitrairement complexes, par exemple qu'un objet contienne des transclusions en push et des transclusions en pull, ainsi qu'une alternance push-pull-push-... .

45 Hormis la fonctionnalité de marquage (par l'algorithme « push ») des sources obtenues en pull, tant l'algorithme push (invoqué en direction de l'aval lors de modifications) que l'algorithme pull (invoqué en direction de l'amont lors d'accès en lecture) doivent ignorer les transclusions dans le mode opposé.

- La figure 57 montre un exemple un peu] e trois transclusions en cascade dans la situation pull-push. Les objets en trait plein sont modifiés localement, et pas ceux en traitillés. Les flèches en traitillés indiquent les références à obtenir en pull, et les flèches en pointillés indiquent les références à consulter seulement pour savoir si de nouveaux enfants sont apparus (l'inner-text étant obtenu dans la version courante dans ces cas).

Architecture centralisée ou en cascade

Tant l'algorithme pull que l'algorithme push peuvent être mis en œuvre soit par une architecture centralisée (avec un serveur de collaboration) soit par une architecture en cascade.

Voici tout d'abord une description de la méthode centralisée.

- 10 Le serveur de collaboration gère les tables de transclusions et les tables de changements locaux (décrivant les objets différences pour les transclusions).

- 15 Dans le cas de la méthode push, ce sont les serveurs de contenu ou l'extension de l'outil de manipulation d'information qui informent le serveur de collaboration d'un changement à apporter dans ces tables, alors que dans le cas de la méthode pull ces tables fonctionnent comme un cache, le serveur de collaboration allant, à chaque requête, interroger les serveurs de contenu.

Ainsi, l'algorithme s'effectue entièrement sur le serveur de collaboration, et les serveurs de contenu ne communiquent pas entre eux.

La méthode en cascade fonctionne comme suit.

- 20 On n'utilise plus de serveur de collaboration, et ce sont les serveurs de contenus qui chacun gèrent leurs tables de transclusions et de changements locaux. Pour les transclusions en push, les tables de transclusion¹⁰³ indiquent les serveurs à contacter en cas de changement local, alors que pour les transclusions en pull, les tables indiquent les serveurs à interroger lors d'une requête. Autrement dit un serveur de contenu connaît les résultats des transclusions en push dont il contient la source, et les sources des transclusions en pull dont il contient le résultat.

L'architecture centralisée a l'avantage que si un serveur de contenu est inaccessible, les tables de différences stockées dans le serveur collaboratif sont utilisées et l'algorithme fonctionne quand même, au pire avec une version qui n'est pas la plus récente ou avec certains contenus manquants.

- 30 L'avantage de l'architecture en cascade, d'un autre côté, est que le traitement est distribué et peut donc être plus efficace en présence de structures de transclusions complexes.

- 35 On peut également considérer une architecture mixte, avec plusieurs serveurs de collaboration, chacun responsable d'un certain nombre d'objets. Lors de transclusions dont tant la source que le résultat sont gérées par le même serveur de collaboration, la méthode centralisée est utilisée, et à chaque changement de serveur de collaboration, la méthode en cascade est utilisée.

La figure 58 montre un exemple avec trois serveurs de collaboration. Les rectangles sont les serveurs de collaboration, et les petits cercles sont les objets. Les grands cercles regroupent les serveurs de collaboration avec les objets qu'ils traitent. Les flèches montrent des transclusions.

- 40 Les lignes doubles continues représentent les parties de l'algorithme qui travaillent de façon centralisée, alors que les lignes doubles interrompues correspondent aux communications en cascade. Le serveur au milieu donne un exemple de serveur de collaboration dégénéré à qui ne correspond qu'un objet, ce qui représente le cas d'un objet qui ne travaille qu'en cascade.

¹⁰³ Les données en question (les références et les différences) ne sont pas forcément stockées sous forme de tables mais peuvent aussi être dans les contenus eux-mêmes.

Propagation des contributions branches

On va à présent montrer comment intégrer le système de contributions d'objets de remplacement et alternatifs à une structure d'objets contenant des transclusions.

- 5 Pour un objet donné on appelle « source créée » l'objet le plus lointain qui soit accessible via une chaîne de transclusions (implicites ou non). Un objet créé localement est donc sa propre source créée.

Les éventuels objets de remplacement et alternatifs d'un objet auront donc tous la même source créée que l'objet en question.

- 10 On définit une relation d'ordre (de création) des objets ayant la même source créée. Si ces objets n'ont pas été créés simultanément, il s'agit de l'ordre de création (on entend par création l'instant où l'objet a commencé à être accessible, même s'il n'avait qu'une existence implicite).

Lorsque plusieurs objets sont créés simultanément, dans le cas de transclusions interne, l'ordre des nouveaux objets correspondra à l'ordre de leurs sources déjà existantes.

- 15 La figure 59 donne un exemple. Les nombres indiquent l'ordre de création des enfants. Cet ordre a été obtenu de la façon suivante.

- 20 Tout d'abord seuls existaient les objets a et 1. Ensuite a été effectuée la transclusion dont le résultat est b. Implicitement ceci a causé la création de 2. Ensuite a été créée la transclusion 3 dont la source est 1. Ceci a provoqué l'apparition implicite de 4. 3 et 4 sont donc apparus simultanément, et on applique donc la règle que l'enfant 1 existant avant l'enfant 2, le correspondant de 1 est considéré comme étant apparu avant le correspondant de 2.

Ensuite est effectuée la transclusion dont le résultat est c. Ceci provoque l'apparition simultanée de 5 et 6. Ici encore, l'ordre 5<6 est choisi pour correspondre à l'ordre de leurs sources immédiates 1<3.

- 25 Cet ordre étant établi, le fonctionnement des remplacements d'objets est le même qu'expliqué précédemment, c'est-à-dire que, dans un ensemble d'objets ayant la même source créée, le plus élaboré est celui qui a été créé le plus tard, et qui fournit un contenu local. Si un objet est modifié localement mais qu'au moins un objet plus élaboré ayant la même source créée avait été créé plus tard et est incompatible avec le contenu modifié¹⁰⁴, alors le premier objet plus élaboré (et les suivants dans la même branche) est (sont) déplacé(s) dans une nouvelle branche (Règle 7).¹⁰⁵
- 30

Variante

Dans cette autre méthode on ne tient pas compte de l'ordre de création des objets mais plutôt de la structure de transclusions.

- 35 Premièrement, on ne change pas des objets de branche (ou crée de branche) tant qu'il n'y a pas de refus explicite (d'objet de remplacement).

L'objet de remplacement pour un objet donné est le contenu plus récent ayant la même source créée (ceci a l'avantage sur l'autre méthode que deux utilisateurs peuvent alternativement contribuer à un même objet tout en restant dans la même branche).

¹⁰⁴ Voir la description des grains.

¹⁰⁵ Autre façon de voir : dans l'ensemble des objets certains spécifient explicitement dans quelle branche ils sont, alors que les autres suivent implicitement tous les déplacements de leur prédécesseur. Ainsi, lorsqu'un objet incompatible doit être placé dans une autre branche seul à ce dernier sera imposé une sélection explicite de branche, et les suivants le suivront implicitement.

- Lorsqu'un utilisateur refuse un objet de , deux situations différentes peuvent se produire, en fonction du chemin à suivre pour atteindre l'objet remplacé depuis l'objet de remplacement, en suivant les relations de transclusion. Si le chemin va en direction de l'amont de l'objet remplacé, alors ce dernier se déplace dans une nouvelle branche. Si au contraire le
- 5 chemin va en direction de l'aval de l'objet remplacé (donc que l'objet de remplacement est un descendant de l'objet remplacé), alors c'est l'objet de remplacement qui est déplacé dans une nouvelle branche.

Ces changements de branche n'altèrent bien sûr pas les relations de transclusion existantes.

La figure 60 illustre trois cas de figure.

- 10 La structure en arbre représente la structure de transclusion (chaque objet est image par transclusion de celui représenté au-dessus de lui, et tous les objets, avant le refus, sont dans la même branche).

- 15 Les cercles hachurés représentent l'objet qui effectue le refus (l'objet remplacé) et les cercles pointillés sont l'objet refusé (l'objet de remplacement). Les lignes traitillées indiquent la région qui sera déplacée dans une nouvelle branche.

Lorsqu'un objet se déplace ainsi dans une nouvelle branche, tous ses enfants qui ne spécifient pas explicitement la branche dans laquelle ils souhaitent être suivent leur parent dans la même branche, et cela continue récursivement avec leurs enfants.

Comportement des grains en présence de transclusions

- 20 Revenons maintenant à l'approche précédente. Pour voir comment la transclusion s'articule avec le procédé déjà décrit de « Remplacement/Enrichissement de Grains »¹⁰⁶, on va maintenant étendre sa structure de données (les arbres de modification de grains), afin de représenter également les transclusions. On va illustrer le fait que les dérivations de ressources reviennent à transclure tout leur contenu ; on va aussi illustrer le cas de transclusion manuelle effectuée dans une ressource que l'on dérive.
- 25

- La figure 63 étend le « Cas 2 » de l'exemple présenté dans la section « Remplacement/Enrichissement de Grains ». Les objets de cette figure sont introduits dans les figures 61 et 62. La figure 61 montre que, dans la ressource R, le grain a est maintenant inclus dans un objet d et, au sein de ce dernier, il est reproduit par transclusion (le grain a est
- 30 transclus vers la droite dans la figure 43).¹⁰⁷

La figure 62 présente le fait que R1 (il s'agit de la ressource R de la figure 61) est dérivée en R2 et que cette dernière est dérivée en R3. Ces deux dérivations donnent lieu respectivement aux transclusions « 2 » et « 3 ». Ainsi la transclusion « 2 » de d résulte en 2:d et la transclusion « 3 » de 2:d lui-même résulte en 3:2:d.

- 35 Par ailleurs, cette même figure 62 présente la transclusion « 1 » du grain a au sein de l'objet d, ce qui donne lieu aux transclusions implicites « 2:1 » et « 3:2:1 » au sein de 2:d et 3:2:d respectivement.

Les identifiants des objets contenus dans d, 2:d et 3:2:d sont préfixés en fonction de la transclusion qui leur a donné naissance.

- 40 Comme on l'avait déjà décrit, les objets qui sont le résultat de plusieurs transclusions dont au moins une est implicite ont plusieurs sources possibles. C'est toujours à partir de la source la plus proche que l'on propagera vers eux les changements.

¹⁰⁶ Voir plus haut la section « Remplacement/Enrichissement de Grains ».

¹⁰⁷ Ceci nous permettra d'avoir des transclusions implicites dans l'exemple.

Dans la figure 63, les objets hachurés rep qui ont été modifiés localement (et qui
sont donc identifiés dans la table des changements locaux). On suppose les étapes suivantes :

1. d dans R2 et c dans le grain **a3** (dans R3) ont été modifiés en premier – c prend ainsi la valeur c3, et d prend ainsi la valeur d2. Ainsi d2 et **a3** deviennent les plus élaborés¹⁰⁸.
5 A cette étape, c3 du grain **a3** se propage vers le grain **3:2:1:a3**. Par ailleurs, dans R1, l'objet a1 (l'innertext du nœud racine du grain **a1**) est modifié en a4 et l'objet 1:a1 est modifié en a5 dans le grain **1:a1** (ce dernier est le grain transclus de **a1**, toujours dans R1). Ces modifications se propagent respectivement vers **a2** puis **a3** et vers **2:1:a2** puis **3:2:1:a3**.¹⁰⁹
- 10 2. Ensuite b1 et c1 dans le grain **a2** (dans R2) sont modifiés en b2 et c2, et dans R3, d2 est modifié en d3. En conséquence des changements (b2 et c2) dans le grain **a2**, comme ces changements sont incompatibles avec le changement c3 déjà existant dans le grain **a3** de R3, ce dernier (**a3**) est refusé d'office (Règle 7) et se retrouve donc dans une autre branche (dirigée vers la droite dans la figure 63). Du coup, le grain **a3** et le grain **a2** sont tous deux les plus élaborés (dans leurs branches respectives), ce qui est représenté dans la figure 63 par un astérisque. En revanche, les modifications b2 et c2 de **a2** se propagent bien dans le grain **2:1:a2**, mais à partir de ce dernier ils ne peuvent pas se propager à **3:2:1:a3**, car c3 qui existe déjà dans **3:2:1:a3** (par propagation de **a3** dans l'étape 1) a une source plus proche que la source de c2. Par ailleurs d3 est aussi le plus élaboré (présenté aussi avec un astérisque).
- 15 3. Dans R2, dans le grain **2:1:a2**, c est modifié localement pour prendre la même valeur c3 que dans **a3**. En conséquence c3 de **3:2:1:a3** (et provenant de **a3**) ne représente plus un obstacle à la propagation de b2 (qui représente une source plus proche) du grain **2:1:a2** vers le grain **3:2:1:a3**. A noter que, du point de vue de l'arbre des modifications d'objets (en ce qui concerne le marquage des objets les plus élaborés) le grain **3:2:1:a3** est maintenant implicite et c'est le grain **3:2:1:a3** qui est le plus élaboré (marqué d'un astérisque dans la figure 63).
- 20
- 25

La figure 63 présente la situation après la dernière étape (c'est-à-dire après l'étape 3 ci-dessus).

- 30 Dans la figure 63, les objets couverts de points représentent les objets qui se trouvent dans la table des changements provenant de l'amont (et qui sont donc modifiés en résultat de propagation de changements locaux).

Dans la figure 63 la distance que représente chaque transclusion a été indiquée entre crochets.

- 35 Comme le serveur collaboratif ne mémorise pas les valeurs des objets mais seulement leurs identifiants, dans le serveur collaboratif on peut associer à ces identifiants un code (checksum) par la méthode Cyclic Redundancy Check (CRC) connue en soi ou une méthode analogue. Ainsi, dans la figure 63, dans l'étape 3 de l'exemple donné ci-avant, la valeur c3 dans **2:1:a2** permet de propager les valeurs de ce grain vers le grain **3:2:1:a3**, grâce au fait que le checksum de c3 de **2:1:a2** est égal à celui de c3 du grain **a3**.

40 **Compteurs d'activité**

Afin de mesurer l'« activité »¹¹⁰ des ressources et de certains des objets qu'elles contiennent, des compteurs sont notamment associés à chaque (ou certains) nœud(s)

- représentant un objet dans un arbre de modification d'objets,

¹⁰⁸ Comme expliqué précédemment, seul des grains (ou des objets n'appartenant à aucun grain) peuvent être « le plus élaboré ».

¹⁰⁹ Ces chemins sont pris car la propagation va suivre le chemin le plus court, comme expliqué plus loin (les distances de chaque transclusion sont indiqués entre crochets dans la figure).

¹¹⁰ Quantité ou fréquence d'accès par les utilisateurs, de dérivations, de transclusion, etc.

- représentant une ressource dans une transclusion de ressources,
- supplémentaire associé à une ressource, représentant un objet non-modifiable contenu dans cette ressource.

Ces compteurs servent notamment à mesurer

- 5 • le nombre de fois qu'un objet donné créé ou transclus dans une ressource se trouve dans une ressource dérivée de cette dernière et/ou
- le nombre de fois qu'un objet donné créé ou transclus dans une ressource est présenté au sein de cette ressource ou d'une ressource dérivée de cette dernière et/ou
- le nombre de fois qu'un objet donné est transclus, ou encore
- 10 • le nombre de fois qu'un groupe d'objets comprenant un objet donné et tous ses descendants (à l'aval), dans l'arbre de modification d'objets qui lui correspond, est transclus par dérivation ou simplement consulté ,
- le nombre de fois qu'une ressource donnée est dérivée ou simplement consultée,
- etc.

15 **Objets conteneurs, Transclusion-fusion, Placement automatique**

Les figures 64 à 88 présentent des opérations de transclusion dans le cadre d'un explorateur de fichiers, en l'occurrence dans Windows Explorer.

Rappelons que chaque objet est potentiellement un conteneur¹¹¹ dans le sens où il peut contenir d'autres objets (objets enfants). On va maintenant considérer

- 20 • qu'une transclusion a¹¹² comme effet :
 - soit la création d'un nouvel objet qui va contenir le résultat de cette transclusion (comme on l'a décrit jusqu'ici)
 - soit la « fusion » de son résultat avec un objet pré-existant qui contient éventuellement lui-même des objets conteneurs préexistants ;
- 25 • et qu'à chaque conteneur peuvent être spécifiées des contraintes (ou préférences) par rapport aux propriétés des objets qu'il peut recevoir.

- Lors d'une transclusion en mode fusion d'un objet source vers un objet destinataire, la valeur (innertext) de l'objet source remplace celle de l'objet (nœud racine) destinataire et ses enfants sont automatiquement placés sélectivement dans les conteneurs contenus dans l'objet destinataire et dont les contraintes associées sont compatibles avec les propriétés desdits objets enfants. Optionnellement, dans le cas où aucun objet conteneur destinataire n'existe ou ne peut recevoir un objet provenant de la source¹¹³, un conteneur « ramasse-reste » est créé et ledit objet provenant de la source est placé directement dans ce dernier.

- La figure 89 illustre une transclusion-fusion de répertoires (objets conteneurs typiques) qui sont réorganisés de manière à ce que les contenus des répertoires A et C, ainsi que les contenus des répertoires eux-mêmes contenus dans A, aillent dans le répertoire A', que le contenu de B aille dans B' et que tous les autres objets aillent dans C'.¹¹⁴

¹¹¹ (« container » en terminologie anglo-saxonne)

¹¹² (selon le mode de transclusion choisi par l'utilisateur qui effectue la transclusion)

¹¹³ (car ayant des contraintes qui sont incompatibles avec lui)

¹¹⁴ Ainsi cette figure présente un exemple où les noms des répertoires sont des propriétés (dont il est question dans les contraintes associées aux conteneurs qui se trouvent à la destination de la transclusion). D'autres propriétés telles que la date de création ou de dernière modification peuvent par exemple aussi représenter de telles propriétés.

La figure 90 présente un exemple où :⁵⁶ de placer automatiquement, dans un document contenant des objets conteneurs (tels que les cellules d'une feuille de calcul de tableur), des liens vers les documents contenus dans la source d'une transclusion, en fonction des propriétés de ces derniers.¹¹⁵

5 L'avantage de telles transclusions est que lorsqu'un nouvel objet est ajouté à la source il est automatiquement propagé au bon emplacement à la destination ; lorsqu'une propriété d'un objet est changée à la source, il peut être déplacé d'un conteneur destination à l'autre (selon les contraintes associées à ces derniers) ; et lorsqu'une contrainte d'un conteneur destination est modifiée, son contenu peut être déplacé vers un autre conteneur destination.

10 Dans la suite on appelle « domaine » d'un conteneur, l'ensemble de tous les objets possibles ayant des propriétés compatibles avec l'ensemble de contraintes associées à ce conteneur. Et on appelle domaine d'un objet l'ensemble des valeurs compatibles avec ses propriétés.

15 Une métrique est automatiquement associée à la largeur du domaine de chaque conteneur, et dans le cas où plusieurs conteneurs peuvent recevoir un objet en question, le conteneur choisi est celui dont le domaine est le plus petit.

Par exemple, on peut adopter une métrique S définie ainsi :

- $S(p) = \psi$ où $0 < \psi < 1$, et 'p' est une propriété atomique
- $S(\sim P) = 1 - S(P)$ P, P₁, P₂ sont ensemble de propriétés
- $S(P_1 \cap P_2) = S(P_1)S(P_2)$
- 20 • $S(P_1 \cup P_2) = S(P_1) + S(P_2) - S(P_1)S(P_2)$
- $S(P_1 - P_2) = S(P_1)(1 - S(P_2))$

Les conteneurs enfants héritent des propriétés de leurs parents et ajoutent (par $\cap$) leurs propres propriétés.

Lors d'une transclusion en mode fusion, le procédé de transclusion consiste à

- 25 1. rechercher pour chaque objet¹¹⁶ source tous les conteneurs destination dont le domaine est égal ou plus large que celui de l'objet source en question ;
2. dans le cas où il y en a plusieurs, utiliser une métrique (décrite ci-dessus) pour mesurer la grandeur de chaque domaine et déterminer le conteneur destinataire dont le domaine est le plus petit ;
- 30 3. propager l'objet source dans ledit conteneur le plus petit ainsi sélectionné.

Les objets enfants peuvent ainsi être placés automatiquement dans l'objet destinataire d'une transclusion.

35 En sens inverse, le domaine d'un conteneur peut être élargi avec les propriétés des objets que l'utilisateur y place « manuellement ». Cette méthode permet de donner aux conteneurs les propriétés des objets qu'on y insère, sans pour autant mentionner ces propriétés explicitement. C'est donc en quelque sorte une méthode très simple d'apprentissage automatique. Autres options : Avant d'élargir le conteneur destinataire, l'utilisateur se voit présenter l'option de restreindre le domaine de l'objet placé. Enfin, dans le cas où l'objet appartient à une « classe » (au sens taxonomique) différente de la classe du conteneur, on peut rendre la classe de l'objet

40 sous-classe de la classe du conteneur. Le placement d'objet devient ainsi un outil d'aide à la création d'une taxonomie.

Une autre manière (ou une manière complémentaire) de procéder est de baser l'apprentissage sur le contenu des objets insérés (e.g. par apprentissage Bayésien naïf). Dans le cas où

¹¹⁵ Cet exemple met en plus en œuvre une transformation qui remplace chaque document source par un lien vers lui.

¹¹⁶ Dans le cas d'un grain composé de plusieurs conteneurs, ceux-ci seront pris ensemble.

l'apprentissage ne correspond pas au ⁵⁷'utilisateur, ce dernier peut corriger l'apprentissage en plaçant ailleurs un objet placé automatiquement. L'apprentissage basé sur le contenu est complémentaire à l'apprentissage basé sur les propriétés du contenu dans la mesure où le premier sera utilisé lorsque le second ne peut pas l'être ou lorsque le second mène à une ambiguïté.

Enfin, un troisième procédé consiste à rétro-propager les objets, c'est-à-dire à les propager en sens inverse de la transclusion-fusion. Ceci est nécessaire notamment lorsqu'une modification représentant un objet de remplacement ou alternatif doit être propagé en amont. Ceci s'effectue de la manière suivante :

- 10 Lors du placement d'un objet, le procédé comporte une étape supplémentaire consistant à mémoriser une association avec le conteneur source. Lorsque tous les objets d'un conteneur destination proviennent d'un même conteneur source, tout nouvel objet ajouté dans le conteneur destination sera automatiquement rétro-propagé vers le même conteneur source. Dans les autres cas, sélectionner un conteneur source qui inclut le domaine de l'objet destination à rétropropager et/ou qui a l'intersection la plus grande avec le domaine de l'objet destination à rétropropager. Et dans le cas où le domaine du conteneur qui a l'intersection la plus grande n'inclut pas le domaine de l'objet destination à rétropropager, l'élargir de manière à l'inclure. Enfin, on peut aussi placer l'objet rétro-propagé dans un conteneur ramasse-reste comme évoqué plus haut.

20 **Conditions associées aux transclusions**

Le propriétaire d'un objet (sous-objets inclus) peut contraindre les transclusions possibles à partir de cet objet. Lorsque l'utilisateur tente de transclure un objet source, des conditions ou options à choisir telles que parmi les suivantes peuvent lui être présentées :

- L'objet ne peut pas être transclus
- 25 • L'objet (ou un ou certains de ses objets enfants) ne doit pas être modifié
- L'objet peut être transclus mais ne doit pas être republié
- L'objet peut être transclus et modifié mais une autorisation de republication doit être adressée au propriétaire avant de le republier et
 - 30 ○ Il pourra être republié (en attendant la réponse) tant que la réponse n'est pas un refus
 - Il ne pourra être republié que seulement dès qu'une réponse d'acceptation sera parvenue – à noter que dans ce cas la loi permet dans certains cas qu'une courte citation soit publiée ; on mettra en œuvre un système automatique pour une telle publication et son remplacement automatique par la publication normale dès que l'acceptation arrive le cas échéant.
 - 35 • L'objet peut être transclus et modifié, mais les contributions faites à l'aval doivent être automatiquement proposées au propriétaire et d'ores et déjà ce dernier a le droit de les republier.
 - Lorsqu'une contribution aval est ainsi acceptée,
 - 40 ▪ le propriétaire et l'aval peuvent la publier
 - seul l'aval peut la publier
 - seul l'amont peut la publier.
 - Lorsqu'une contribution aval est refusée,
 - ni l'aval ni le propriétaire peuvent publier ladite contribution

- seul le propriétaire ladite contribution
- l'aval peut publier ladite contribution.

En outre les mêmes conditions peuvent ou doivent être reproposées plus à l'aval.

5 Le propriétaire à l'aval de la transclusion peut contraindre les publications à l'amont. Ainsi, lorsque la modification d'un objet (sous-objets inclus) remonte pour être présentée à l'amont, le propriétaire amont peut se voir proposer notamment la ou les conditions ou options à choisir parmi les suivantes :

- L'objet ne peut pas être modifié
- La modification présentée ne peut être republiée
- 10 • Republication (sans modification) autorisée mais seulement après émission d'une requête d'acceptation
 - et après réception de l'acceptation
 - ou en attendant cette acceptation et tant qu'un refus n'est pas reçu.
- 15 • La contribution de l'aval peut être encore modifiée à l'amont, mais republiée par l'amont seulement après émission par l'amont d'une requête d'acceptation de la part de l'aval
 - et après réception de l'acceptation
 - ou en attendant cette acceptation et tant qu'un refus n'est pas reçu.
- 20 • Lorsqu'une re-modification par l'amont est acceptée par l'aval,
 - l'amont et l'aval peuvent la publier
 - seul l'amont peut la publier
 - seul l'aval peut la publier
- Lorsqu'une re-modification par l'amont est refusée par l'aval, l'aval peut quand même la republier.

25 En outre les mêmes conditions peuvent (ou doivent) être reproposées plus à l'amont.

Architecture de personnalisation

30 Les transformations et conditions décrites dans les sections précédents, ainsi que les autres procédés décrits jusqu'ici (transclusion, remplacement d'objets, présentation d'objets alternatifs...) peuvent rentrer dans un cadre général de « personnalisation » que l'on va maintenant décrire. On utilise le terme « personnalisation » puisque ces conditions et transformations peuvent être spécifiées¹¹⁷ comme étant fonction du ou des (groupes d') utilisateurs auxquels ils s'adressent.

35 On définit la personnalisation comme une fonction qui sur la base d'objets¹¹⁸ (contenu et position) et/ou des méta-données qui leur sont associées ou encore d'autres données de l'environnement, ainsi qu'en fonction de groupes d'utilisateurs ciblés, retourne le contenu et la position des mêmes ou d'autres objets. Une personnalisation peut aussi prendre en entrée le résultat d'une autre personnalisation.

¹¹⁷ (par le propriétaire d'une ressource ou par défaut selon une spécification pré-établie ou encore selon d'autres moyens)

¹¹⁸ Chaque objet peut notamment être identifié par un identifiant ou par sa position dans la structure de la ressource.

Un mécanisme de chiffrement permet que les conditions faisant partie des personnalisations sont respectées. Par exemple, dans le cas où une ressource peut être dérivée on va s'assurer qu'un certain objet (tel qu'une publicité) qu'elle contient ne puisse pas être modifié. On va aussi s'assurer que ces conditions vont être héritées d'une transclusion à l'autre.

La figure 91 montre l'architecture d'un système mettant ceci en œuvre. On y distingue un ensemble nommé Serveur de Personnalisation, lui-même composé des serveurs suivants :

- Un Serveur Collaboratif qui gère notamment les transclusions et les autres procédés décrits dans ce document ;
- Un Serveur de Clés qui mémorise toutes les clés pour les contenus cryptés et qui est chargé de communiquer avec le Serveur d'Authentification pour authentifier l'utilisateur ;
- Un Serveur d'Authentification (qui peut être par exemple un serveur tel qu'Active Directory de Microsoft).

La figure 91 montre aussi la partie cliente (plug-in) qui comprend

- Un Client de Personnalisation. C'est un composant qui est notamment chargé de soumettre les « credentials » (login/password) au Serveur de Clés, d'obtenir du Serveur Collaboratif les personnalisations et de les appliquer aux objets spécifiés par l'Extension d'Outil de Manipulation ; de crypter (lors d'une publication) les objets désirés, et de décrypter de manière transparente les objets cryptés.
- L'Extension d'Outil de Manipulation qui a notamment la charge de fournir une interface utilisateur permettant de choisir une personnalisation à appliquer ; prendre de l'utilisateur ses credentials et les communiquer au client de Personnalisation ; obtenir les contenus à partir de serveurs de contenu ; dans le cas où un contenu est publié, transmettre le contenu et les credentials de l'utilisateur au Serveur de Clés.

A noter que la partie cliente (plug-in) est authentifiée par le Serveur de Clés et le Serveur Collaboratif comme condition à leur fonctionnement.¹¹⁹

Dans la figure 91, les étapes du procédé correspondent aux numéros indiqués :

1. L'Extension d'Outil de Manipulation obtient le contenu de la version courante à partir du Serveur de Contenu correspondant
2. L'Extension d'Outil de Manipulation demande au Client de Personnalisation la personnalisation qui doit être appliquée à la ressource, en lui transmettant le contenu de la ressource (qui devra être personnalisé/décrypté) et les credentials de l'utilisateur

¹¹⁹ Pour ce faire, pour chaque installation de la partie cliente on utilise une paire de clés publique-privée. La clé publique est stockée dans le Serveur de Clés et la clé privée l'est dans le Client de Personnalisation. Pour assurer la sécurité de la clé privée dans ce dernier, on pourra utiliser un code généré à partir des spécifications des composants matériels essentiels de la machine (disque dur, processeur, MAC si possible, etc.) transformés au moyen d'une fonction de hachage (telle que Blowfish). En surplus, lors du démarrage, le Serveur de Personnalisation peut vérifier le checksum (ou autre empreinte) du code exécutable de la partie cliente.

En premier, la partie cliente obtient l'information d'enregistrement (telle que le propriétaire du logiciel, etc.) et utilise le code généré (à partir des spécifications des composants matériels essentiels de la machine) pour lire sa clé privée, puis crypte l'information d'enregistrement avec la clé privée et l'envoie au Serveur de Clé pour authentification dans une session. Le Serveur de Clé décrypte l'information en utilisant la clé publique qu'il a dans sa base de données et vérifie les données de checksum et d'enregistrement pour ce client. En cas de succès un canal sécurisé est alors établi entre la partie cliente et serveur en utilisant un protocole tel que SSL.

3. Dans le cas où des objets : Client de Personnalisation requiert les clés au Serveur de Clés
4. Le Serveur de Clés vérifie les credentials de l'utilisateur en contactant le Serveur d'Authentification
5. Dans le cas d'authentification réussie le Serveur de Clés transmet les clés au Client de Personnalisation
6. Dans le cas d'authentification réussie le Client de Personnalisation demande au Serveur Collaboratif les personnalisations qui doivent être appliquées à la ressource
7. Le Client de Personnalisation demande les objets requis (déterminés à l'étape 6) de leurs Serveurs de Contenus respectifs
8. Le Client de Personnalisation applique alors les personnalisations, en décryptant les contenus cryptés. Le résultat est transmis à l'Extension d'Outil de Manipulation qui présente le résultat.

15

Dans le cas où dans le Serveur de Contenu, le contenu d'une ressource telle qu'un document est générée dynamiquement à partir de requête(s) à une base de données (ou autre type d'organisation de données), le procédé sera mis en œuvre de manière à ce que l'utilisateur puisse manipuler et collaborer au niveau de la ressource qui lui est présentée et que les modifications résultantes soient répercutées dans la base de données (de manière transparente).

20

La difficulté réside dans l'identification de tels objets mis à la disposition de l'utilisateur. S'il s'agit d'une base de données relationnelle, on va considérer chaque ligne de la base donnée et chaque champs (chaque colonne de la table en question) comme présenté dans la figure 92. Pour chaque ligne, l'identifiant de l'« objet » (au sens de la présente description) en question pourra par exemple être la clé primaire. De même l'identifiant d'objet pour les champs pourra être la composition de la clé primaire et du nom de la colonne de la table en question.¹²⁰

25

Un composant (plug-in) supplémentaire est alors nécessaire au niveau du Serveur de Contenu pour injecter, au moyen de balises « invisibles » dans la présentation (telle que <A> en HTML), l'information de l'identification de l'objet correspondant à la donnée de la base de données. Ainsi toute requête au Serveur de Contenu (destinée in fine à la base de données) est en fait adressée à ce composant et ce dernier manipule la requête de manière à ce que le résultat en retour contienne en plus l'identifiant d'objet de chaque donnée retournée par la base de données.

30

Ainsi on peut identifier les objets. Lorsque par exemple un objet 'o' d'une base de données est réutilisé dans une pluralité de ressources (dont le contenu est dynamiquement généré à partir de cette dernière), lorsque l'une de ces ressources R1 sera dérivée en R2 et l'objet 'o' modifié, l'Extension d'Outil de Manipulation en question va informer le Serveur de Collaboration de cette dérivation et de la modification de 'o'. Lorsque l'une des ressources parmi ladite pluralité de ressources va être re-visionnée au moyen d'un Outil de Manipulation, son Extension demandera¹²¹ au Serveur de Collaboration les remplacements ou enrichissements à effectuer. Ce dernier lui retournera alors l'identifiant de la modification (si elle est toujours la plus élaborée) de 'o' dans R2. L'Extension obtiendra alors cette version modifiée de 'o' à partir du Serveur de Contenu hébergeant R2.

35

40

¹²⁰ L'identification d'objets dans une base de données relationnelles est connu en soi.

¹²¹ (en suivant les étapes décrites ci-avant)

Remplacements/Enrichissement d'acceptation

e propositions

Le propriétaire d'une ressource peut accepter ou refuser les objets de remplacement et les objets alternatifs présentés dans cette ressource.

- 5 Dans le cas où un objet (de remplacement ou alternatif) est refusé, tous les autres objets de la branche dans laquelle cet objet se trouve sont implicitement aussi refusés.

On va maintenant présenter une interface utilisateur mettant en œuvre le procédé. L'outil de manipulation d'information illustré ici (en guise d'exemple) est un navigateur Web doté d'une extension apte à assembler des objets de remplacement et alternatifs dans les pages Web. Les
10 mêmes principes peuvent être mises en œuvre sur tout autre outil de manipulation d'information et pour manipuler d'autres types de ressources.

La figure 93 présente schématiquement le navigateur Web muni d'une barre d'outil (qui représente l'extension en question). L'utilisateur peut ainsi entrer une URL
15 « www.site.org/page » et cliquer sur « Envoyer » pour faire afficher à l'écran la page Web correspondant à cet URL, dans le cadre d'un ERC « E1 » (qu'il peut sélectionner dans une liste déroulante).¹²²

La figure 94 présente la page Web en question affichée, dans le cadre de l'ERC, avec des objets de remplacement ou alternatifs (schématiquement représentés par des rectangles), ainsi qu'un bouton « Modifier » dans la barre d'outil.

- 20 La figure 95 présente schématiquement le navigateur après que l'utilisateur ait cliqué sur le bouton « Modifier » et qu'il ait procédé à des modifications. Ce bouton est maintenant remplacé par un bouton « Enregistrer ». Lorsque l'utilisateur clique sur ce dernier, dans le cas où il est le propriétaire de la page en question, le système lui donne le choix d'enregistrer les modifications soit dans le même fichier, soit en tant que nouvelle page dérivée dans un fichier
25 différent à indiquer ; dans le cas où l'utilisateur n'est pas le propriétaire de la page en question, il doit nécessairement l'enregistrer en tant que nouvelle page dérivée.¹²³

La figure 96 présente schématiquement que l'utilisateur a la possibilité d'accepter ou refuser les objets de remplacement et alternatifs. L'utilisateur peut accepter/refuser un tel objet en se positionnant dessus et

- 30
- en le sélectionnant puis en cliquant dans « Accept » ou « Reject » dans la barre d'outil
 - ou en cliquant sur un des symboles «  » ou «  » présentés (affichés seulement quand le pointeur de la souris est positionné sur l'objet en question). A noter que les flèches (triangles) orientés vers la gauche et la droite servent à faire présenter respectivement l'objet alternatif précédent et suivant (à commencer par les objets originaux, puis les
35 objets de remplacement). Ces derniers sont présentés ressource par ressource (i.e. les objets provenant d'une même ressource sont présentés ensemble). A noter que lorsqu'un objet alternatif est modifié, il peut être considéré comme « gelé », les autres options n'étant alors plus accessibles via ces flèches (mais une opération « undo » permettant de revenir à l'état précédent)

40

 - ou encore en sélectionnant « Accept » ou « Reject » dans le menu contextuel qui apparaît lorsqu'il presse le bouton droit de la souris.

¹²² Alternativement, le bouton « Envoyer » n'est pas nécessaire si l'envoi se fait de manière classique en utilisant le navigateur Web et si pour ne pas visionner la page à travers un ERC on peut par exemple sélectionner « pas de ressources collaboratives » dans la liste déroulante.

¹²³ Lorsqu'il enregistre en tant que nouvelle page dérivée, il devient bien sûr propriétaire de cette dernière.

Alternativement (et avantageusement), l'un objet alternatif peut être déterminé automatiquement lorsque l'utilisateur enregistre une ressource dérivée dans laquelle l'objet en question a été remplacé par ledit objet alternatif ; et le refus d'un objet de remplacement peut être déterminé automatiquement lorsque ce dernier a été remplacé par l'objet original (par exemple en utilisant la flèche orientée vers la gauche mentionnée dans le paragraphe précédent).

L'interface utilisateur permet en outre de mettre en évidence les objets (réifications) et les grains, ou encore d'en créer.

Modes d'utilisation des ERC

- 10 Les modes d'utilisation des ERC concernent leur activation ou pas (pour propager les mises à jour vers l'aval à travers les transclusions ainsi que les autres contributions).

L'interface utilisateur permet aux utilisateurs de sélectionner d'abord des ERC par défaut, puis pour chaque accès à une ressource de préciser les modes respectifs des ERC (toutefois ces modes persistent par défaut d'un accès à un autre).

- 15 On offre à l'utilisateur un écran comme suit comportant deux parties :

Première partie :

☒ ☐ (case à cocher pouvant prendre l'une parmi deux valeurs : cochée ou pas) –

Coché voulant dire « Par défaut, les mises à jour sont pleinement propagées (à partir de l'amont et vers l'aval) »

- 20 Pas coché voulant dire « Par défaut, les mises à jour sont reçues de l'amont mais pas propagées vers l'aval » (mode secret)

Deuxième partie :

Pour chaque ERC mentionné par l'utilisateur :

☒ ☒ (case à cocher pouvant prendre l'une parmi deux valeurs : coche ou croix) –

- 25 Coche : entière collaboration dans cet ERC

Croix : aucune communication dans cet ERC

On va maintenant présenter l'utilisation de ces modes au moyen de scénarios.

Les figure 97 présente une ressource R2 R1 dans l'ERC E1 et qui transclut (par la transclusion T2) un objet 'c' de la ressource S1 dans les ERC E2 et E3.

Dans la figure 98 la ressource R2 est ouverte par l'utilisateur dans un mode « secret » pour l'ERC E3, ensuite T2:d est modifié.

5

Dans la figure 99 la ressource R2 est ouverte dans l'ERC E2 et T2:d est refusé.

Ainsi l'utilisation de modes est utile pour collaborer sélectivement¹²⁴. On verra plus loin une variante (dans la section présentant les objets multi-versions).

Recommandation de Ressources et d'ERC

- 10 On a jusqu'ici que l'utilisateur peut accéder à une ressource « à travers » des ERC. Une question qu'il peut se poser est de savoir, parmi les ERC qu'il a le droit d'utiliser, lesquels choisir.

Une autre question de l'utilisateur peut être de savoir quelles sont les ressources les plus récentes et/ou pertinentes pour lui que d'autres utilisateurs ont vues dans les ERC qu'il est en train d'utiliser.

15

Les deux questions ci-dessus peuvent être combinées : l'utilisateur peut demander au Serveur Collaboratif de lui fournir à la fois des ressources pertinentes (et/ou récentes) et des ERC parmi ceux qu'il a le droit d'utiliser qui soient pertinents pour aller visiter lesdites ressources pertinentes.

- 20 Enfin les utilisateurs peuvent aussi vouloir se découvrir entre eux (par exemple pour se rencontrer ou encore s'échanger des informations) à partir des ressources/ERC qu'ils vont respectivement visiter ou manipuler (voir aussi à ce sujet l'approche de rating des utilisateurs décrite à la section « Procédé de tri d'objets alternatifs par notation (rating) »).

- 25 Dans la mesure où le Serveur Collaboratif sait quel utilisateur accède à quelle ressource dans quel ensemble de ressource collaborative (ces triplets étant mémorisés), il peut simplement appliquer des méthodes de filtrage collaboratif pour obtenir les réponses aux quatre types de questions ci-dessus, notamment en effectuant des comptages pour déterminer la proximité des utilisateurs en termes de nombre de couples 'ressources/ERC' qu'ils ont visités/utilisés en commun et pour échanger entre utilisateurs proches leurs (bonnes) expériences respectives.

- 30 La proximité des utilisateurs entre eux (en termes de nombre de couples 'ressources/ERC' qu'ils ont visités/utilisés en commun) peut bien entendu aussi servir en complément de l'approche de rating susmentionnée, pour trier les objets alternatifs en fonction de la proximité de leurs auteurs par rapport à l'utilisateur.

- 35 On verra plus loin une variante basée sur la mise en œuvre de conduits de propagation de scores de pertinence.

Avantages

On peut maintenant présenter des avantages résultant des procédés décrits ici. On va aussi décrire quelques nouveaux procédés associés.

¹²⁴ L'utilisation d'ERC dans différents modes est utile lors de contributions de modification de résultats de requêtes persistantes à une base de données (comme déjà décrit, une requête persistante peut être vue comme une transclusion). Lorsque l'utilisateur ajoute un élément dans les résultats d'une requête, les critères de sélection qui figurent dans la requête persistante en question servent à rétro-propager dans la base de données les propriétés associées aux données. Ceci s'effectue exclusivement dans les ERC concernés.

Avantages pour les procédés tion

L'ERC représente un moyen de collaboration efficace pour rédiger/composer une œuvre à plusieurs contributeurs. Il suffit pour cela que tous les contributeurs partagent¹²⁵ un même ERC pour que chaque contributeur puisse *élaborer* davantage les ressources produites par les autres.

Pour publier, les propriétaires des ressources qui seront publiées *acceptent* ou *sélectionnent* (parmi les objets alternatifs) les contributions qu'ils souhaitent voir publier (et ils ignorent ou refusent les autres). Ainsi les autres contributions ne sont pas vues des utilisateurs autres que ceux qui sont membres dudit même ERC.

- 10 Non seulement de nouvelles pages peuvent ainsi être publiées, mais même des pages créées à l'extérieur de l'ERC peuvent être dérivées et republiées, sous réserve des considérations légales décrites plus loin.

Rédaction collective d'une publication future

- 15 Une application intéressante est la préparation d'une publication future. Son principe est illustré à la figure 100. Les rectangles (R1, R11, etc.) représentent des ressources ; les flèches représentent des dérivations ; les régions en forme de « L » à l'envers, encadrées par des lignes traitillées, représentent des ERC (E1, E2 et E3).

- 20 Il s'agit ici de la structure de l'organisation de la préparation de la publication¹²⁶ d'une revue mensuelle, dont le modèle de présentation est la ressource « Template ». Par exemple, cette revue doit être publiée la première fois en Avril (ressource R1), puis en Mai (ressource R2), puis encore en Juin (ressource R3), etc.

Les collaborateurs de la parution d'Avril sont membres de l'ERC E1 ; ceux de la parution de Mai sont membres de E2 ; et ceux de Juin de E3.

- 25 Ces ensembles de collaborateurs (respectivement membres de E1, E2 et E3) peuvent être distincts ou non, et ces ensembles de collaborateurs travaillent en parallèle pour préparer les parutions disons 6 mois à l'avance.

- 30 La parution d'Avril (R1) est dérivée de la ressource « Template », de manière à réutiliser ce modèle. Les collaborateurs chargés de produire le contenu de R2 (Mai) reprennent la présentation de la parution d'Avril. Ceux chargés de produire la parution de R3 (Juin) reprennent la présentation du contenu de R2.

L'avantage de cette approche est double :

- (par les propriétés de la transclusion) toute mise à jour faite à l'amont se répercute à l'aval ; ainsi par exemple si le modèle Template est amélioré, toutes les parutions en profitent puisqu'elles s'en trouvent à l'aval,
- 35 • et réciproquement toute proposition d'élaboration soumise (de l'aval) par les collaborateurs d'une parution ne sera vue des collaborateurs de la parution ultérieure (qui forment une autre branche à l'aval) que si elle est acceptée. En effet, les collaborateurs d'une parution R ne voient pas les contributions des collaborateurs de la parution R-1 avant qu'elles ne soient acceptées, puisqu'ils ne sont pas membres du
- 40 même ERC.

Bien entendu toutes sortes d'autres organisations des dérivations peuvent être envisagées, pour offrir divers avantages selon les applications souhaitées.

¹²⁵ (soient utilisateurs de)

¹²⁶ (c'est-à-dire de la structure du « workflow » de publication)

Avantages pour diffuser des contenus publicitaires

On va maintenant décrire comment un éditeur ou autre organisme, appelons le « Le journal », peut faire de la publicité sur l'Internet en utilisant les procédés décrits ici.

5 Le journal crée un nouvel ERC (Le journal peut aussi utiliser un ERC préexistant). Dans son site, un lien cliquable permet de télécharger une extension (plug-in) aux outils de manipulation d'information courants tels que les navigateurs courants Internet Explorer ou Firefox. Dès ce téléchargement et suite à l'installation automatique de ladite extension, ledit ERC est activé d'office dans l'outil de manipulation d'information de l'utilisateur. Dans le cas où l'extension est déjà installée, l'ERC est directement activé sans qu'il y ait besoin de téléchargement ni d'installation. Tout ceci n'est effectué que par un simple clic de l'utilisateur.

10 Un contenu (objet) publicitaire est inclus dans chaque ressource (page Web) et une contrainte de non-modifiabilité lui est associée (voir plus loin les sections « Le Web en lecture-écriture » et « Association de contraintes à des objets »). Ainsi à chaque fois qu'une page du site est dérivée par un utilisateur et que cette page est republiée à son tour, les visiteurs de cette page dérivée voient ledit objet publicitaire qui, du fait de la contrainte de non-modifiabilité, ne peut être supprimé ou modifié en utilisant les outils de manipulation d'information munis de l'extension (plug-in) en question.

20 Dans les cas courants où les publicités sont placées dans des <IFRAME> les procédés de comptage classiques peuvent continuer à fonctionner. En tout état de cause, des compteurs tels que ceux mentionnés à la section « Mise en œuvre globale » pourront être mis en œuvre pour compter le nombre de fois qu'une nouvelle page est dérivée en incluant l'objet publicitaire, le nombre de fois qu'il est transclus par d'autres moyens, ou encore le nombre de fois où l'objet publicitaire est simplement vu dans une page. Ceci permet au journal de rentabiliser ces publications en faisant payer leurs annonceurs en fonction de ces compteurs.

25 Par ailleurs, le fournisseur du service rendu par le serveur collaboratif (pour gérer l'ERC qui incluent les objets publicitaires en question) est rétribué par Le journal, ou son client annonceur ou encore l'agence insérant les publicités, en fonction des résultats donnés par ces mêmes compteurs.

30 En outre, un autre procédé consiste à ce que l'outil de manipulation des informations *filtre* le contenu des objets non-modifiables (comme les publicités) dans le cas où la valeur du compteur de transclusions de¹²⁷ cet objet dépasse un certain seuil (seuil qui bien évidemment peut être fonction du paiement déjà effectué ou prévu d'effectuer au fournisseur du service rendu par le serveur collaboratif).

Avantages pour les « wikis »

35 Les « wikis » (cf. www.wikipedia.com) sont des sites dont les visiteurs peuvent modifier les pages à souhait. Les wikis sont particulièrement bien adaptés pour échanger des idées et les articuler et les développer en commun. Ils sont typiquement utilisés pour coopérer sur des projets collectifs, encourager les contributions et accélérer la circulation des informations.

40 Toutefois, dans les wikis, les parties « sensibles » (sujettes à débat) des contenus sont en général modifiés et re-modifiés alternativement par les « utilisateurs de chaque camp » jusqu'à qu'éventuellement un consensus (sur ce que doivent comporter ces informations) émerge... Il est ainsi difficile de mettre des informations « non neutres » dans les wikis. Bien que les wikis présentent l'avantage d'encourager à respecter les opinions d'autrui, la quête de neutralité inhérente aux wikis paraît trop restrictive et tend parfois à appauvrir l'information.

¹²⁷ (et/ou de dérivation de ressources contenant)

On propose ici une sorte de nouveau type de wikis, non seulement permettez mais même encouragez la contribution d'informations non-neutres (insertion de contenus provenant de points de vue divergents, d'opinions différentes, etc).

- 5 Les différents mécanismes (arbres de dérivation, transclusion, etc.) décrits jusqu'ici permettent de présenter les mises à jour comme des objets de remplacement (OR) et les informations non-neutres comme des objets alternatifs (OA).¹²⁸

Le Web en lecture-écriture

- 10 Les modifications faites par chaque internaute dans n'importe quelle page Web sont visibles par chaque autre internaute qui la visite dans le cadre d'un même ERC. Cependant, pour respecter les « copyright », il serait normal que seules les modifications autorisées par les propriétaires des pages Web puissent être republiées via les serveurs collaboratifs. Or, comme évoqué plus haut, on peut s'attendre que les propriétaires de pages Web aient intérêt à (et donc veuillent bien) autoriser les modifications de leurs pages dans la mesure où ils augmentent
- 15 ainsi leur audience et ils peuvent contraindre les pages de manière à ce que certaines parties (par exemple les publicités qu'elles contiennent) ne puissent pas être modifiées. Dans la mesure où ils le feront, le Web actuel qui est en lecture seule se transformera en un Web en lecture-écriture.

- 20 Une autre différence essentielle avec les wikis actuels est qu'ici les ressources d'un même ERC ne sont pas forcément dans un même site. De plus les wikis actuels ne comportent que des pages web, alors qu'ici on vise la collaboration sur toutes sortes de ressources accessibles sur un réseau.

- 25 En résumé, le Web qui aujourd'hui est en lecture seule peut devenir en lecture-écriture. De plus on peut ajouter à ce nouveau Web les ressources autres que des pages Web qui sont dans les ERC.

Liens inverses

- Un lien ajouté, par un utilisateur d'un ERC, dans une ressource dérivée à partir d'une ressource (telle qu'une page Web) de départ, peut amener un utilisateur (le même ou un autre) du même ERC à visiter la ressource pointée par ce lien ajouté.
- 30 On met en plus en œuvre un procédé consistant à ajouter automatiquement, sur une ressource dérivée de ladite ressource pointée, un lien (« lien inverse ») pouvant amener vers ladite ressource de départ un utilisateur du même ERC visitant ladite ressource pointée.

- L'ensemble des liens inverses peut par exemple être automatiquement présenté à l'utilisateur dans une (sous-)fenêtre, adjacente à la (sous-)fenêtre présentant le contenu de la ressource visitée.
- 35

Méta-données de contrainte de diffusion

- Le fait que techniquement n'importe quelle ressource accessible, par exemple n'importe quelle page Web, puisse être republiée, incite à mettre en place des aides pour se mettre en règle, au moins sur le plan légal. L'idée essentielle est que l'outil de manipulation d'informations automatise ou facilite l'émission d'une requête pour demander l'autorisation de (re)publier un
- 40 contenu, puis gère le processus de publication en attendant la réponse à une telle requête.

¹²⁸ (tout en évitant les redondances dans le stockage des informations)

Ainsi, l'activation de moyens automatique⁶⁷ de d'autorisation peut être déclenchée dès qu'une ressource est dérivée¹²⁹ en vue d'être publiée, et la publication démarrera automatiquement au plus tard quand l'autorisation arrivera.

5 En attendant une autorisation de publier, un mécanisme permettant de ne publier que des citations (extraits restreints) prises dans des limites raisonnablement (légalement) permises pourra être mis en œuvre.

10 Pour qu'un tel procédé de demande d'autorisation puisse être mis en œuvre, il faut que le propriétaire du contenu d'origine, ou une autorité décisionnaire (qui puisse décider à sa place), puisse être contacté. On propose ainsi que les ressources incluent ou associent des méta-données de contact d'une autorité décisionnaire.

15 D'autres méta-données peuvent être associées à un objet pour sélectivement contrôler sa modification et/ou son accessibilité par des tiers au sein de ressources dérivées. Notamment, des méta-données de contraintes de diffusion peuvent spécifier que (comme déjà évoqué) des objets donnés peuvent être reproduites (avec modification autorisée ou non) *à condition* que des sous-parties spécifiées soient incluses sans modification.

20 La ressource contenant –ou à laquelle sont associées– des méta-données peut être structurée sous forme d'une arborescence d'objets, de manière à ce qu'en association avec au moins certains nœuds de l'arborescence, une méta-donnée de contrainte de diffusion soit mémorisée, et qu'une méta-donnée de contrainte de diffusion au niveau d'un nœud s'applique aux nœuds enfants du nœud en question.

Avantage de contourner la censure

25 Lors de l'accès à une ressource, si l'utilisateur n'a pas expressément spécifié au moins un ERC (et/ou un serveur collaboratif) pour cet accès, l'outil de manipulation d'information envoie une requête au(x) serveur(s) collaboratif(s) configuré(s) par défaut, pour obtenir les identifiants des objets de remplacement et des objets alternatifs.

Pour répondre à une telle requête, chaque serveur collaboratif est apte à rechercher d'autres serveurs collaboratifs qui peuvent fournir les identifiants requis, et à retourner ces identifiants¹³⁰ à l'outil de manipulation d'information.

30 L'outil de manipulation d'information est ainsi apte à obtenir les identifiants requis à partir de n'importe quel serveur collaboratif, ce dernier servant d'intermédiaire s'il ne peut répondre lui-même. Il n'est ainsi pas utile de bloquer, notamment dans un but de censure, l'accès à¹³¹ certains serveurs collaboratifs puisque d'autres pourront les relayer.

35 La figure 101 illustre ce procédé : au lieu de recevoir les identifiants des objets de remplacement et des objets alternatifs à partir d'un serveur collaboratif ES1, car la connexion avec ES1 est coupée (ce qui est représenté par le gros X dans la figure 101), l'outil de manipulation d'information les obtient à partir d'un serveur collaboratif ES2 avec lequel la connexion est possible. En effet, lorsqu'il reçoit la requête de l'outil de manipulation d'information, ES2 interroge les serveurs collaboratifs qu'il connaît, ceux-ci relayent à leur tour la requête aux serveurs qu'ils connaissent eux-mêmes, et ainsi de suite, dans un procédé
40 classique en peer-to-peer connu en soi et que l'on ne décrit donc pas ici davantage. ES2 finit

¹²⁹ Voir aussi la section « Conditions associées à une transclusion ».

¹³⁰ Le Serveur collaboratif est apte à retourner les identifiants d'objets de remplacement et d'objets alternatifs, ainsi que leurs identifiants d'ERC respectifs, de manière à ce que les identifiants d'objets de remplacement et d'objets alternatifs puissent être utilisés sélectivement, c'est-à-dire seulement si les ERC qui les fournissent conviennent à l'utilisateur ou qu'il en soit membre)

¹³¹ (aux adresses IP de)

ainsi par atteindre ES1 qui fournit les i⁶⁸
alternatifs pour la ressource accédée en question.

objets de remplacement et des objets

- 5 Par ailleurs, il n'est pas utile non plus de bloquer l'accès à des sites de contenu (PCS ou SCS) dans la mesure où non seulement les informations qu'ils fournissent sont reproduites dans les « caches » des pages dérivées à partir d'elles, mais en plus les serveurs collaboratifs sont aptes à fournir la page dérivée accessible la plus proche (au sens de proximité de dérivation) d'une page requise.

Extensions aux protocoles de syndication actuels

Interopérabilité

- 10 On va maintenant présenter deux protocoles (PSE et ASE) qui sont des extensions aux protocoles existants de « partage » (ou syndication), tels que RSS. Ces protocoles ont en principe les caractéristiques suivantes :

- Le contenu est publié sous forme de canaux (« channel » ou « feed » en terminologie anglo-saxonne)
- 15 • Un canal comprend plusieurs « items » (éléments d'information)
- Chaque item a un identifiant unique
- Des serveurs peuvent s'abonner aux canaux publiés par d'autres serveurs
- Des serveurs (« re-publieurs ») peuvent agréger plusieurs canaux et les republier

- 20 Certains de ces protocoles (tels que « RSS+SSE » spécifié par Microsoft) permettent aussi que :

- les re-publieurs puissent modifier certains items (c'est-à-dire ces serveurs permettent à leurs utilisateurs de modifier les items dont ils ne sont pas les auteurs d'origine)
- les résultats de ces modifications soient propagés aux (serveurs) abonnés et s'y synchronisent de manière décentralisée.

- 25 Les protocoles PSE et ASE que l'on va introduire visent ainsi un réseau de serveurs qui partagent des contenus en s'abonnant à des canaux publiés par d'autres serveurs. Dans un environnement hétérogène, les serveurs utilisent des protocoles différents. Par exemple certains utilisent RSS 1, d'autres RSS 2.0 avec l'extension SSE (RSS+SSE), d'autres encore peuvent utiliser encore d'autres extensions, etc. Les protocoles décrits ici sont en fait des extensions à tous protocoles existants et futurs qui ont les caractéristiques énoncées ci-dessus ;
- 30 ils ont été conçus pour fonctionner comme des couches additionnelles à eux.

On exprime les protocoles de partage existants au moyen d'une syntaxe abstraite¹³² comme suit :

- La balise <channel> pour les canaux qu'un serveur publie ;
- 35 • La balise <item> (dans la balise <channel>) pour chaque item dans le canal publié – elle possède l'attribut 'id' qui lui donne un identifiant ;
- La balise <content> (dans la balise <item>) pour le contenu – par exemple cette balise correspond tant aux balises <title> que <description> de RSS 2.0 ;¹³³

- 40 Chaque protocole existant qui possède une représentation équivalente (c'est-à-dire des balises et attributs équivalents) peut interopérer avec PSE et ASE¹³⁴. Toute autre balise que celles

¹³² Les balises et les attributs (en XML) présentés sont « abstraits » en ce sens qu'ils correspondent aux différentes balises/attributs des protocoles existants. En réalité ce seront les balises et attributs des protocoles existants qui seront utilisés et non pas ceux-ci.

¹³³ Comme il s'agit ici de syntaxe abstraite, en réalité on n'aura pas cette balise <content> mais les balises <title> et/ou <description>, etc. ou balises analogues, selon le protocole d'origine en question.

¹³⁴ Dans la suite par « PSE » on entend « PSE et ASE » sauf sens contraire manifeste.

équivalentes à celles ci-dessus seront
pas interprétés) par le protocole PSE.

es mais ignorées (c'est-à-dire ne seront

La mise en œuvre de PSE va typiquement inclure la mise en œuvre des protocoles existants les plus courants tels que RSS 2.0, RSS 2.0 avec SSE, RSS 1.0 etc. de manière à ce que leurs balises (et attributs) respectives soient interprétées dans leurs protocoles d'origine respectifs. Lorsque la mise en œuvre de PSE ne reconnaît pas une balise elle l'ignore mais la retient dans le canal de manière à ne pas perdre d'information pour les autres serveurs qui participent au réseau.

Le protocole PSE met d'abord en œuvre les spécifications du (ou des) protocole(s) de la source (ou d'un protocole choisi par l'utilisateur par défaut dans le cas où le serveur est la source lui-même) et puis ajoute les éléments PSE (comme frères ou enfants d'éléments « item ») et effectue les traitements associés. Les éléments PSE sont déclarés dans un espace de nommage (« namespace » en terminologie anglo-saxonne) distinct, et l'on prévoit qu'ils seront simplement ignorés par les autres serveurs qui ne les reconnaîtraient pas (c'est-à-dire qui ne traite pas les éléments appartenant audit espace de nommage). En général, une copie des éléments de la source est aussi conservée de manière à ce que les serveurs abonnés qui ne comprennent pas PSE puissent quand même interpréter le protocole de la source et interopérer ainsi avec PSE. Comme déjà mentionné, le protocole PSE effectue les traitements propres au(x) protocole(s) source(s)¹³⁵ de manière à ce que le serveur apparaisse (aux serveurs abonnés qui ne comprendraient que le protocole source) comme un serveur du protocole source.

La figure 102 présente différents serveurs publiant du contenu dans différents protocoles. Le serveur qui met en œuvre PSE peut interopérer avec ces divers protocoles et ajouter l'extension PSE aux canaux respectifs. Par exemple il prend 'Channel 1' de "Server 1" qui utilise RSS 2.0 avec l'extension SSE, ajoute la couche PSE dans 'Channel 1' et le republie. Il publie aussi 'Channel 3' en format OPML 1.0 + SSE (le format peut être choisi par l'utilisateur qui crée le canal) auquel s'abonne "Server 4" qui reconnaît ce protocole.

Cette figure illustre aussi le fait que dans le protocole PSE, le protocole source est retenu afin que ceux qui comprennent le protocole source puisse s'abonner au canal republié. Par exemple 'Channel 1' est republié par "Server 3" en utilisant le même protocole que la source, à savoir RSS 2.0 avec SSE, de manière à ce que "Server 5" qui comprend le RSS 2.0 avec SSE puisse s'abonner à 'Channel 1'. Quant à "Server 7" elle utilise le protocole ASE et peut s'abonner à tous les canaux publiés par "Server 3" mais en ignorant les éléments en PSE.

Bien entendu, l'agrégation de canaux différents n'est possible que pour des protocoles compatibles. Par exemple RSS 1.0 (ou 0.92) peut être agrégé avec RSS 2.0 pour donner un canal RSS 2.0 valide mais ceci n'est malheureusement pas possible dans l'autre sens. Un serveur mettant en œuvre PSE permet d'agréger de tels protocoles compatibles. Ceci est illustré dans la figure 102 où "Server 3" agrège 'Channel 1' et 'Channel 2' comme un canal RSS 2.0 + SSE. D'autres serveurs comme "Server 5" peuvent alors s'y abonner pour y voir des items tant de 'Channel 1' que de 'Channel 2'.

PSE

L'extension PSE aux protocoles de partage permet à l'utilisateur de transclure d'une source un item ou une partie (du contenu) d'un item. L'utilisateur peut marquer une partie¹³⁶ d'un item comme étant un objet ; ce dernier peut alors être transclus dans un autre canal ou dans le même canal. Les caractéristiques décrites jusqu'ici de la transclusion restent valides. Ainsi, les

¹³⁵ Ici protocole source veut dire soit le protocole de la source de l'item republié, soit le protocole choisi par l'utilisateur pour les items originaux.

¹³⁶ (ce qui a pour effet de lui faire acquérir un identifiant ; il s'agit du procédé de réification décrit plus avant)

modifications d'objet, dans le résultat de la transclusion, sont conservées lors de la synchronisation avec la source ; et les sources plus proches sont prioritaires pour les mises à jour.

Spécification XML

- 5 On a juste besoin d'une seule balise qui est définie dans l'espace de nom "pse:". C'est la suivante :

Élément <pse:part> dans <item> ou <channel>

Cet élément peut être dans l'élément <item> ou dans l'élément <channel> au même niveau que <item>. Quand une partie d'un *item* est réifiée¹³⁷, cet élément <pse:part> apparaît dans l'élément <item> pour encapsuler l'objet créé. Lorsqu'un élément <item> est transclus dans un autre canal au même niveau qu'un élément <item> alors il est inséré dans l'élément <channel>. A noter que la position de cet élément compte.

Attributs:

- 15 • Tid: Ceci est un attribut *requis* qui donne l'identifiant de l'objet selon le protocole PSE.
- Src-tid: Ceci est un attribut *optionnel* qui donne le Tid de la source modifiée la plus proche. Lors des synchronisations, le mécanisme de mise à jour de cet attribut est le suivant : l'attribut src-tid est comparée avec (le cas échéant) l'attribut src-tid de l'élément source afin de prendre sa valeur si elle est plus proche. Dans le cas où il n'y a pas d'attribut src-tid, cela signifie que la source ne dépend pas elle-même d'une autre source.
- 20 • Changetype: Ceci est un attribut *optionnel* qui est ajouté quand l'objet est modifié localement. Il peut prendre sa valeur parmi les suivantes : CREATED, MODIFIED, DELETED.

Scenarios PSE

Les scénarios qui suivent couvrent différents cas d'utilisation de PSE.

Item transclus dans un canal

La figure 103 présente deux flux (canaux) RSS, *Server 1* et *Server 2*, et le fait qu'un item de *Server 1* est transclus pour être placé entre deux items existants de *Server 2*.

Représentation XML abstraite

Comme déjà spécifié, le nouvel élément <pse:part> possède les attributs tid, src-tid et changetype. Le tid est l'identifiant de l'objet en question qui, dans ce scénario, est le résultat de la transclusion. Le 'src-tid' est l'identifiant de la source modifiée la plus proche. Seul l'un parmi les attributs src-tid et changetype seront présents car quand l'objet est modifié localement il n'y a plus de source.

Server 1 (la source):

```

40   <item id=11>
       <content>
           ...
       </content>
   </item>
   <item id=12>

```

¹³⁷ (pour être considérée comme un objet)

```

        <content>
            ...
        </content>
    </item>
    <item id=13>
        <content>
            ...
        </content>
    </item>
10    <item id=14>
        <content>
            ...
        </content>
15    </item>

```

A noter qu'il peut y avoir d'autres éléments (tels que `<sx:sync>`) dans `<item>` (non mentionnés dans un but de concision) qui sont ignorés dans le protocole PSE proprement dit mais retenus pour l'usage du protocole d'origine.¹³⁸

En résultat de la transclusion, *Server 2* comprend l'item '13' transclus ainsi qu'un élément

`<pse:part tid="T1:13" src-tid="13"/>` (initialement sans contenu).

Server 2:

```

    <item id=21>
        <content>
            ...
        </content>
    </item>
    <pse:part tid="T1:13" src-tid="13"/>
    <item id=13>
30    <content>
        ...
    </content>
    </item>
    <item id=22>
35    <content>
        ...
    </content>
    </item>

```

'13' est modifié dans *Server 1*

Lorsque *Server 2* se synchronise avec *Server 1*, seul l'item '13' dans *Server 2* est mis à jour avec le nouveau contenu de sa source¹³⁹ et, tant que le contenu de l'élément 'T₁:13' n'a pas été modifié par l'utilisateur (localement), l'élément `pse:part` reflète implicitement le contenu de l'item '13'.

'T₁:13' est modifié dans *Server 2*

Lorsque 'T₁:13' dans *Server 2* est (localement) modifié¹⁴⁰, son contenu est explicitement ajouté dans `<pse:part>`, et la modification est effectuée dans l'item '13' située dans *Server 2*

¹³⁸ Le Server PSE inclut non seulement l'implémentation de PSE mais aussi les implémentations des autres protocoles qu'il comprend (tel que par exemple RSS 2.0 et SSE). Ainsi, bien que ces éléments soient ignorés par le protocole PSE, ils sont pris en compte par l'implémentation de leur protocole original respectif dans le Server PSE lui-même.

¹³⁹ (en utilisant le protocole d'origine)

¹⁴⁰ Avantagusement, la mise en œuvre du procédé comprend des moyens pour déterminer par exemple semi-automatiquement (ou simplement en le demandant à l'utilisateur) si la modification en question est dépendante du contexte ou pas. Le fait de conserver l'item '13' ou non en sera alors lié. Ici « dépendante du contexte » signifie essentiellement que la modification cite au moins une partie d'un autre item du même canal.

pour l'usage d'autre protocole. Toute mise à jour ultérieure sera faite dans 'T₁:13' et dans '13'. L'attribut 'changetype="modified"' est ajouté et l'attribut 'src-tid' est supprimé :

Server 2:

```

5      <item id=21>
        <content>
          ...
        </content>
      </item>
10     <item id=13>
        <content>
          ... ← contenu modifié
        </content>
      </item>
15     <pse:part tid="T1:13" changetype="modified">
        ... ← contenu modifié
      </pse:part>
      <item id=22>
        <content>
20          ...
        </content>
      </item>

```

L'item '13' continuera à se synchroniser avec sa source (par le protocole d'origine), alors que l'élément 'T₁:13' ne recevra plus les mises à jour de la source puisqu'il est modifiée localement.

Item transclus au sein d'un autre item

La figure 104 présente une transclusion dont le résultat est placé au sein du contenu d'un autre item.

30

Représentation XML abstraite

L'élément `pse:part` est maintenant placé au sein du contenu de l'item '22' (pour représenter le résultat de la transclusion à l'endroit précis où ce résultat est placé). Comme dans l'exemple précédent, la copie du contenu de l'item source '13' est en plus ajoutée dans *Server 2*.

Server 1:

```

35     <item id=11>
        <content>
          ...
        </content>
      </item>
40     <item id=12>
        <content>
          ...
        </content>
      </item>
45     <item id=13>
        <content>
          ...
        </content>
      </item>
50     <item id=14>
        <content>
          ...

```

```

        </content>
    </item>

```

Server 2:

```

5      <item id=21>
        <content>
            ...
        </content>
    </item>
10     <item id=22>
        <content>
            ...
            <pse:part tid="T1:13" src-tid="13"/>
            ...
        </content>
15    </item>
        <item id=13>
            <content>
                ...
            </content>
20    </item>
        <item id=23>
            <content>
                ...
            </content>
25    </item>

```

'13' est modifié dans Server 1

Similairement à l'exemple précédent, lorsque *Server 2* se synchronise avec *Server 1*, l'item '13' dans *Server 2* est mis à jour avec le nouveau contenu de sa source et l'élément `pse:part` ('T₁:13') s'y réfère.

30 'T₁:13' est modifié dans Server 2

Lorsque 'T₁:13' dans *Server 2* est modifié localement, le contenu modifié est mis dans l'élément `pse:part`, et la modification est effectuée dans l'élément `item` (pour l'usage d'autre protocole). Toute modification locale ultérieure sera faite dans 'T₁:13' et dans '13'. Un attribut 'changetype="modified"' est ajouté et l'attribut 'src-tid' est supprimé dans l'élément `pse:part` :

Server 2:

```

        <item id=21>
            <content>
                ...
            </content>
        </item>
        <item id=22>
            <content>
                ...
                <pse:part tid="T1:13" changetype="modified">
                    ... ← contenu modifié
                </pse:part>
                ...
            </content>
50    </item>
        <item id=13>
            <content>
                ... ← contenu modifié
            </content>
55    </item>
        <item id=23>

```

```

        <content>
            ...
        </content>
    </item>

```

5 Partie d'item transcluse au sein d'un autre item

La figure 105 présente le cas de la transclusion d'une partie seulement du contenu d'un item, cette partie ('113') étant maintenant l'objet source à transclure au sein du contenu d'un autre item ('22').

Représentation XML abstraite

- 10 La seule différence par rapport à l'exemple précédent est que la partie en question dans la source doit être *réifiée*. On utilise pour cela la même balise `pse:part` qui sert ici à délimiter la partie réifiée et dont l'attribut `tid` fournit l'identifiant de l'objet qui résulte de cette réification.

Server 1:

```

15   <item id=11>
        <content>
            ...
        </content>
    </item>
20   <item id=12>
        <content>
            ...
        </content>
    </item>
25   <item id=13>
        <content>
            ...
            <pse:part tid="113">
                ...
            </pse:part>
            ...
        </content>
    </item>
30   <item id=14>
        <content>
            ...
        </content>
    </item>
35   <item id=15>
        <content>
            ...
        </content>
    </item>

```

- 40 Ici l'élément `pse:part` n'a pas d'attribut '`src-tid`' puisque son contenu ne dépend pas d'une autre source.

Bien évidemment, ce qui a été décrit jusqu'ici est aussi valable pour les transclusion dans un même canal ou entre différents canaux d'un même serveur.

Transclusions internes et implicites

- 45 La figure 106 présente le cas d'une transclusion interne dans *Server 1* qui donne lieu à une transclusion implicite¹⁴¹ dans *Server 2*.

Server 1:

```

    <item id=11>
        <content>

```

¹⁴¹ (pour la description de ces termes voir notamment la section « Transclusion implicite, transclusion interne » plus avant)

```

    ...
    </content>
  </item>
  <item id=13>
    <content>
      ...
      <pse:part tid="112">
        ...
      </pse:part>
10      ...
      <pse:part tid="T1:112" src-tid="112">
        ...
      </pse:part>
15      ...
    </content>
  </item>
  <item id=14>
    <content>
      ...
20    </content>
  </item>

Server 2:
  <item id=21>
    <content>
25      ...
    </content>
  </item>
  <item id=13>
    <content>
30      ...
      <pse:part tid="T2:112" src-tid="112">
        ...
      </pse:part>
35      ...
      <pse:part tid="T2:T1:112" src-tid="112">
        ...
      </pse:part>
      ...
    </content>
40  </item>
    <pse:part tid="T2:13" src-tid="13"/>
    <item id=23>
      <content>
        ...
45      </content>
    </item>

```

‘112’ est modifié dans *Server 1*

Lorsque *Server 1* se synchronise avec lui-même (ce qui doit normalement être déclenché directement), ‘T₁:112’ est mis à jour avec le contenu de 112, sa source modifiée la plus proche
 50 (donnée par l’attribut ‘src-tid="112"’). Egalement, lorsque *Server 2* se synchronise avec *Server 1*, les contenus de ‘T₂:112’ et ‘T₂:T₁:112’ dans *Server 2* sont mis à jour avec le nouveau contenu de l’item ‘112’ de *Server 1*, leur source modifiée la plus proche étant également donnée par l’attribut ‘src-tid="112"’,¹⁴².

¹⁴² Les mécanismes de détermination des sources les plus proches ont déjà été décrits plus avant, notamment dans la section « Algorithmes en Push ».

'T₁:112' est modifié dans Server 1

- Quand 'T₁:112' dans *Server 1* est modifié, un attribut 'changetype="modified"' est ajouté à l'élément `pse:part` et l'attribut 'src-tid' est supprimé dans cet élément. Lorsque *Server 2* se synchronise avec *Server 1*, le contenu de 'T₂: T₁:112' dans *Server 2*, tant qu'il n'a pas été
- 5 modifié localement, est mis à jour avec le nouveau contenu de l'item 'T₁:112' de *Server 1*, sa source modifiée la plus proche étant maintenant 'T₁:112'.

'T₂:112' est modifiée dans Server 2

- Comme dans les exemples précédents, lorsque 'T₂:112' dans *Server 2* est modifié, le contenu modifié est mis tant dans l'élément `pse:part` 'T₂:13' que dans l'élément `item` '13'. Un
- 10 attribut 'changetype="modified"' est ajouté dans l'élément `pse:part` 'T₂:112' et l'attribut 'src-tid' est supprimé.
- Lorsque *Server 2* se synchronise avec lui-même¹⁴³, le contenu de 'T₂:T₁:112' est mis à jour avec la source 'T₂:112' puisque sa distance est maintenant plus courte que celle de 'T₁:112'.

ASE

- 15 On va maintenant présenter une extension (« ASE ») aux protocoles de partage qui permet à l'utilisateur de voir des contributions concurrentes (alternatives) à partir de multiples flux RSS auxquels il est abonné ou à partir de flux RSS qui se trouvent dans la « chaîne » d'abonnements dans laquelle il se trouve. On va ainsi appliquer, aux protocoles de partage existants, l'idée d'encourager les points de vue différents (et non pas systématiquement
- 20 remplacer chaque contribution quand une plus récente¹⁴⁴ apparaît).

Les objectifs (supplémentaires à ceux de PSE¹⁴⁵) sont :

- D'accéder aux contributions (objets alternatifs) correspondant à des points de vue (branches) différents, via les canaux auxquels on est abonné (et qui se trouvent donc à l'amont) ;
- 25 • optionnellement de présenter même les contributions qui proviennent de l'aval (contrairement aux protocoles existants), et ceci même dans le cas où il n'y a pas d'abonnements réciproques¹⁴⁶ ou de cycles dans les chaînes d'abonnement. Pour cela, on va utiliser un mécanisme de notification (par lequel les modifications faites sur les items qui sont à l'aval sont communiqués vers l'amont). Bien entendu, ces
- 30 notifications ne sont pas nécessaires dans le cas d'abonnements réciproques.

Les notifications sont seulement faites aux sources immédiates, qui eux-mêmes propagent à leurs propres sources et ainsi de suite. On a ainsi pas besoin d'envoyer les notifications à toutes les sources possibles¹⁴⁷.

Spécification XML

- 35 Dans l'espace de nommage "ase:" les balises XML incluent les suivantes¹⁴⁸ :

¹⁴³ (ce qui doit normalement être déclenché directement par cette modification)

¹⁴⁴ Le protocole RSS+SSE prend encore d'autres facteurs en considération, mais toujours dans cette logique de remplacement systématique.

¹⁴⁵ (notamment à l'objectif de garder les changements locaux (ici dénommés « Original ») qui est en fait un objectif propre à PSE)

¹⁴⁶ Il y a abonnements réciproques lorsque la source s'abonne au flux qui s'est abonné à elle.

¹⁴⁷ (voir aussi plus loin l'apport des Serveurs Collaboratifs)

¹⁴⁸ On ne rappellera pas ici les mécanismes de remplacement et d'enrichissement d'objets. Voir notamment les sections Procédé de création d'objets les plus élaborés et Procédé de détermination de l'objet de remplacement et des objets alternatifs et les suivantes.

Elément <ase:contributions> <rel>

Cette balise se place au plus haut niveau, c'est-à-dire au même niveau que la balise *item*. Elle est présente pour chaque *item* qui (ou si une partie de l'item en question) a des contributions.

Attributs:

- 5
- ID: l'identifiant de l'item dont les contributions sont incluses ;
 - Branch: l'identifiant de la branche active de l'item ;
 - SeqID: numéro de séquence d'un item dans une chaîne d'abonnements.

Elément <ase:contribution> dans <ase:contributions>**Attributs:**

- 10
- Kind: Cet attribut prend l'une des valeurs suivantes : "Original", "Replacement" (remplacement, c'est-à-dire l'objet le plus élaboré de sa branche) ou "Alternative" (provenant d'une autre branche);
 - Branch: Identifiant de la branche de la contribution en question;
 - SeqID: Numéro de séquence de la contribution en question;

15 Elément <ase:changes> dans <ase:contributions>

Cet élément contient la liste des changements les plus récents de l'item. Il permet de déterminer l'objet le plus élaboré dans une branche quand l'objet le plus élaboré courant se déplace dans une autre branche.

Elément <ase:change> dans <ase:changes>

- 20 Cet élément contient l'information du changement de l'item en question.

Attributs:

- Branch: l'identifiant de la branche de l'item modifié;
- SeqID: le numéro de séquence de la branche de l'item modifié.

Elément <ase:movebranch> dans <ase:contributions>

- 25 Cette balise permet de propager l'information d'acceptation/refus d'une contribution par une source.

Attributs:

- Branch: l'identifiant de la nouvelle branche de la source.

Exemple

- 30 Le comportement du protocole ASE est illustré dans la figure 107, dans laquelle :
- "(R)" signifie Remplacement, "(A)" signifie Alternative et "(O)" signifie contenu d'Origine ;
 - Une case hachurée représente un objet modifié localement ; une case couverte de points représente un objet modifié suite à un changement à l'amont ; une case blanche (non couverte de motif) représente un objet non modifié ;
 - Une flèche pleine représente la communication d'un objet par l'amont ; une flèche non pleine (traitillée) représente la notification d'un changement vers l'amont.
- 35

Cette figure illustre la formation des branches¹⁴⁹ d'un objet "a" (les branches sont présentées verticalement) en présentant les étapes suivantes :

¹⁴⁹ Les branches sont les suites de dérivations des objets qui constituent une même version alternative, comme décrit en détail plus avant dans ce document.

- Server 2 s'abonne à Server 1 et Se e à Server 2
- Server 2 modifie "a" en a2
- Server 3 modifie "a" en a3
- Server 1 modifie "a" en a1' refusant ainsi a3
- 5 • Server 2 accepte l'alternative a3
- Server 3 accepte l'alternative a1'
- Server 3 modifie "a" en a1''
- Server 1 refuse a1''.

10 Ces exemples sont conformes aux procédés déjà décrits avant dans ce document, à l'exception du fait que, afin de favoriser la compatibilité avec les protocoles existants, la première branche présentée à l'utilisateur est celle qui contient les objets tels que l'auraient présenté les protocoles existants.

Transclusions et Contributions vers l'amont (PSE avec ASE)

15 En général dans les protocoles existants, un item, même s'il a été modifié localement, lorsqu'il se synchronise avec une source qui a elle-même été mise à jour, se remet à jour par rapport à cette source. Ainsi, comme déjà décrit, dans le protocole PSE seul l'élément `pse:part` garde la modification locale (qu'il est nécessaire de garder dans le cadre du procédé de transclusion) et non pas l'élément `item` (vu que l'on ne veut pas perdre sa conformité avec les protocoles existants).

20 Or, dans le protocole ASE, le contenu original est aussi mémorisé dans une balise `<ase:contribution>` ayant l'attribut `Kind="original"`. Ceci représente donc une redondance. Mais, si on fait l'hypothèse que les serveurs qui comprennent le PSE comprennent aussi le protocole ASE, on peut éviter cette redondance en ne gardant le contenu modifié que dans la balise du protocole ASE. A noter que dans cette approche, un serveur qui

25 ne comprend que le protocole ASE (et pas le PSE) voit tout serveur PSE comme s'il était purement un serveur ASE.

On va maintenant illustrer le fonctionnement conjoint de ces protocoles sur la base de l'exemple de la figure 108 où un item '13' d'un flux *Server 1* est transclus au sein d'un item '22' qui se trouve dans un flux *Server 2* cet item '22' étant lui-même transclus dans *Server 3*.

30 Les flèches en trait non plein (traitillés) représentent le procédé de notification vers l'amont selon le protocole ASE.

Exemple

Représentation XML abstraite

Server 1:

```

35   <item id=11>
        <content>
            ...
        </content>
    </item>
40   <item id=12>
        <content>
            ...
        </content>
    </item>
45   <item id=13>
        <content>
            ...
        </content>
    </item>

```

```

    <item id=14>
      <content>
        ...
      </content>
    </item>

```

Server 2:

```

    <item id=21>
      <content>
        ...
      </content>
    </item>
    <item id=22>
      <content>
        ...
        <pse:part tid="T1:13" src-tid="13">
          ...
        </pse:part>
        ...
      </content>
    </item>
    <item id=13>
      <content>
        ...
      </content>
    </item>
    <item id=23>
      <content>
        ...
      </content>
    </item>

```

Server 3:

```

    <item id=31>
      <content>
        ...
      </content>
    </item>
    <pse:part tid="T2:22" src-tid="22"/>
    <item id=22>
      <content>
        ...
        <pse:part tid="T2:T1:13" src-tid="13">
          ...
        </pse:part>
        ...
      </content>
    </item>

```

‘T1:13’ est modifié

50 Ce changement est notifié vers l’amont à *Server 1* de manière à ce que ce dernier inclut la contribution en question qui est en l’occurrence un remplacement :

Server 1:

```

    <item id=11>
      <content>
        ...
      </content>
    </item>
    <item id=12>
      <content>
        ...
      </content>
    </item>

```

```

    <item id=13>
      <content>
        ...
      </content>
    </item>
    <ase:contributions id=13 branch=0 seqid=0>
      <ase:contribution kind="Replacement" branch=0 seqid=1>
        <ase:source url="Server 2" id=T1:13/>
      </ase:contribution>
10  </ase:contributions>
    <item id=14>
      <content>
        ...
      </content>
15  </item>

```

'T₂:T₁:13' est modifié

Ce changement qui se situe dans *Server 3* est notifié vers l'amont d'abord à *Server 2*, puis à partir de *Server 2* vers *Server 1*, de manière à ce que ce chacun inclut la contribution en question :

```

20  Server 1:
    <item id=11>
      <content>
        ...
      </content>
25  </item>
    <item id=12>
      <content>
        ...
      </content>
30  </item>
    <item id=13>
      <content>
        ...
      </content>
35  </item>
    <ase:contributions id=13 branch=0 seqid=0>
      <ase:contribution kind="Replacement" branch=0 seqid=2>
        <ase:source url="Server 3" id=T2:T1:13/>
      </ase:contribution>
40  </ase:contributions>
    <item id=14>
      <content>
        ...
      </content>
45  </item>
Server 2:
    <item id=21>
      <content>
        ...
      </content>
50  </item>
    <item id=22>
      <content>
        ...
      <pse:part tid="T1:13" changetype="modified">
        ...
      </pse:part>
      </content>
55  </item>
60

```

```

    <item id=13>
      <content>
        ...
      </content>
    </item>
    <ase:contributions id="T1:13" branch=0 seqid=1>
      <ase:contribution kind="Remplacement" branch=0 seqid=2>
        <ase:source url="Server 3" id=T2:T1:13/>
      </ase:contribution>
10  </ase:contributions>
    <item id=23>
      <content>
        ...
      </content>
15  </item>
Server 3:
    <item id=31>
      <content>
        ...
      </content>
20  </item>
    <pse:part tid="T2:22" src-tid="22"/>
    <item id=22>
      <content>
25  ...
        <pse:part tid="T2:T1:13" changetype="modified">
          ...
        </pse:part>
30  ...
      </content>
    </item>

```

Il s'agit ainsi de propager les manipulations de branche de proche en proche lors des synchronisations entre flux, sur la base des principes suivants :

- 35 • Une contribution aura l'attribut `kind` « Remplacement » si elle se trouve dans la branche active, ou « Alternative » dans le cas contraire.
- L'objet ayant l'attribut `kind` « Remplacement » est implicitement refusé lorsqu'une autre branche est modifiée par l'utilisateur. Lui-même et tous les objets qui en dérivent seront alors déplacés à une autre branche.
- 40 • Dans le cas où pour un objet une alternative (qui se trouve donc dans une autre branche que la branche active) est acceptée, ledit objet et les objets qui en dérivent et qui se trouvaient dans la branche qui était active, sont déplacés dans la branche de l'alternative acceptée, à l'exception de l'objet à partir duquel la branche acceptée avait été créée ainsi que les objets qui dérivent de ce dernier.

45 **Serveur Collaboratif pour PSE/ASE**

Avantageusement, les différentes variantes du Serveur Collaboratif décrites avant dans ce document peuvent améliorer les performances des serveurs PSE (et ASE). L'idée est d'éviter les propagations des changements de proche en proche (lors des synchronisations entre serveurs complètement décentralisés) et d'utiliser au moins en partie les serveurs collaboratifs

50 comme intermédiaires centraux qui peuvent exploiter les algorithmes efficaces de synchronisation décrits avant. En particulier, les points à considérer sont les suivants :

1. Si un serveur agrège des items à partir de plusieurs sources situés dans des serveurs différents, alors il doit se synchroniser avec chacun d'eux ;

- 82
2. Si plusieurs canaux sont abonné : source, alors chacun doit obtenir le contenu de cette source;
 3. Si dans une chaîne d'abonnements l'un des serveurs tombe, les serveurs suivants ne recevront plus les mises à jour ni de ce serveur ni de ceux qui y sont à l'amont ;
 - 5 4. Enfin, lors de la recherche de la source la plus proche quand il y en a plusieurs, il faut éviter d'interroger toutes les sources.

Respectivement, les réponses apportées par l'approche d'intermédiation via un (ou plusieurs) Serveur(s) Collaboratif(s) sont les suivantes :

- 10 1. Le Serveur Collaboratif peut s'abonner aux différentes sources et notifier le serveur en question. Si en plus lesdites sources utilisent aussi le Serveur Collaboratif, alors elles vont le notifier directement de leurs mises à jour éventuelles ce qui évitera le polling de la part du Serveur Collaboratif.
2. Le Serveur Collaboratif se charge d'obtenir le contenu de la source en question (ici aussi par polling ou par réception de notifications) et de notifier les mises à jour ;
- 15 3. Les abonnés ne dépendent plus de la résistance de la « totalité des maillons de la chaîne ». En effet, dans le cas le plus défavorable seul le serveur qui tombe arrêtera de fournir (sauf bien sûr si un Serveur Collaboratif tombe lui-même mais on peut imaginer qu'il est doté de moyens nécessaires pour assurer sa fiabilité) ;
- 20 4. Le Serveur Collaboratif garde une trace de tous les changements et ne propage en suivant les trancclusions que les changements provenant de sources les plus proches (voir la section « Algorithme en push »).

En résumé, des Serveurs Collaboratifs placés dans un réseau de syndication ont essentiellement pour effet d'économiser la bande passante et de fiabiliser le réseau lorsque des nœuds de ce réseau constituent des « maillons faibles ».

- 25 Dans les figures 109 et 110, on reprend l'approche (adoptée dans toute cette description) selon laquelle les Serveurs Collaboratifs ne possèdent pas les contenus mais seulement leurs identifiants.

La figure 109 présente l'étape consistant à prendre en compte un changement dans une source.

- 30 Ses sous-étapes sont les suivantes :

1. L'information du changement (sans le contenu) est communiqué au serveur de Collaboration
2. Les abonnés communiquent le contenu aux Serveurs de Contenu

La figure 110 présente l'étape consistant à mettre à jour les abonnés.

- 35 Cette étape comprend les sous-étapes sont les suivantes :

1. Le Serveur de Contenu CS1 interroge régulièrement la source
2. CS1 informe le Serveur de Collaboration des changements (s'il y en a)
3. Les mises-à-jour/remplacements/alternatives sont transmis aux abonnés (par notifications)
- 40 4. Les abonnés obtiennent le contenu des mises-à-jour/remplacements à partir des Serveurs de Contenu ; en principe les alternatives sont obtenues sur demande¹⁵⁰.

¹⁵⁰ Dans le protocole ASE, le serveur contient l'URL du flux auquel il est abonné avec l'identifiant de l'objet en question, de manière à pouvoir récupérer les alternatives sur demande de l'utilisateur.

Transclusion avec transfo

La transclusion décrite jusqu'ici peut être schématisé par un flux d'information comme suit. Tout d'abord les informations proviennent de la source de la transclusion. Puis ces informations sont altérées ou réorganisées comme décrites par les différences de la transclusion. Finalement la structure obtenue est retournée à l'utilisateur.

L'extension décrite dans cette section propose d'ajouter une¹⁵¹ *transformation* à ce flux d'information. Il s'agit d'une fonction (par exemple un filtre sur les objets) qui est attachée à la transclusion. Cette transformation est effectuée sur les objets de la source avant l'application des différences de la transclusion. La figure 111 illustre ce transfert d'informations.

Les objets de la source sont, en première étape et avant de pouvoir être soumis à une application des différences, immédiatement soumis à une transformation : dans la figure 111 par transformation un objet a été supprimé et un autre a vu son contenu modifié. La seconde boîte montre la source telle qu'elle est vue par l'algorithme d'application des différences (en « pull » ou en « push » comme déjà décrit). Ce sont ces objets pré-traités qui sont en deuxième étape déplacés, supprimés ou modifiés comme indiqué par les différences qui, dans l'exemple présenté dans la figure 111, consistent en une suppression, une création et un déplacement.

Mise en œuvre

Quelque soit la mise en œuvre choisie pour le système (« push » ou « pull »), le principe de la méthode à appliquer ici est le même. Une transformation est appliquée à chaque fois que le système a besoin du contenu d'un objet dans le cadre d'une transclusion. Par exemple pour la méthode « push », lorsqu'un attribut (comme le contenu) d'un objet modifié est envoyé vers le résultat de la transclusion, il est immédiatement soumis aux transformations, avant d'être intégré au résultat (il peut cependant être avantageux de tout d'abord vérifier s'il sera effectivement utilisé, afin d'éviter l'application de la transformation pour un objet qui ne sera pas visible dans le résultat). De même, pour la méthode « pull », lorsqu'un objet est téléchargé afin d'être intégré au résultat, il devra être soumis à la transformation.

Autrement dit, l'algorithme de transclusion doit fonctionner comme si la source de l'objet était le résultat de la transformation. Par exemple si une transformation supprime un objet alors la transclusion doit fonctionner comme si cet objet n'existait pas à la source. Autre exemple, si la transformation effectue un tri sur les objets de la source alors la transclusion fonctionne comme si les objets étaient déjà triés à la source.

La transformation est toujours la même sur les enfants et sur les parents, pour une même transclusion. Toutefois, on pourra prévoir deux fonctions de transformation au lieu d'une et les appliquer alternativement : l'une au moment de la création d'une transclusion, l'autre à chaque mise à jour (cette dernière pourrait selon les cas nécessiter de connaître l'état précédent de l'objet source).

Enfin, la nature de la transformation appliquée par une transclusion peut être stockée de la même façon que les autres informations (attributs) des objets (« inner-text », position, etc) en association avec les résultats de transclusions¹⁵². Ceci permet de la considérer de manière similaire aux autres attributs vis-à-vis de la Règle 8 « *déterminer la source la plus proche lorsqu'il y a plusieurs sources candidates* » ainsi que de la Règle 9 qui est exposée plus loin.

¹⁵¹ Lorsqu'aucune transformation n'est associée à la transclusion, on dira que c'est la transformation identité qui y est associée.

¹⁵² La transformation d'une transclusion est traitée comme un attribut de l'objet racine du résultat (i.e. de l'unique objet faisant partie du résultat de la transclusion, et contenant tous les autres objets faisant aussi partie du résultat de cette transclusion).

Flux d'information dans le cas de transcl

Il s'agit de savoir quel chemin prend le flux d'information, c'est-à-dire dans quel ordre les transformations sont appliquées. Comme le montre la figure 112, on voudrait que les transclusions les plus intérieures (internes) soient suivies en premier (et les plus extérieures en dernier).

Dans cette figure à gauche en haut se trouve un objet contenant une transclusion interne (r), puis une transclusion (u) de cet objet est effectuée, plaçant le résultat à droite. Ensuite le tout est transclus (s) à nouveau, plaçant le résultat en bas. Les flèches indiquent les directions de flux d'information, avec une lettre représentant la transformation effectuée.

- 10 L'intuition est que la fonction « u », de la transclusion effectuée de gauche à droite, est appliquée sur tous les objets de la source de la transclusion, même si ces derniers sont des résultats de transclusions internes. Dans la source de la transclusion « u », la transformation « r » est effectuée avant la transformation « u » qui lui est plus externe (et cette transformation « r » n'est donc pas effectuée sur l'objet qui se trouve en haut à droite). Par ailleurs on voit
- 15 dans cette figure qu'aucune transclusion implicite se trouvant dans la moitié inférieure de la figure (dans le résultat de s) n'est suivie. On va donc devoir concevoir des règles qui, dans la moitié supérieure de la figure, pour relier l'objet supérieur gauche à l'objet inférieur droit, ferois passer par le coin inférieur gauche. Et si on considère l'ensemble de cette figure, pour relier l'objet supérieur gauche à l'objet le plus en bas à droite, elles devront faire passer par
- 20 l'objet correspondant en haut à droite plutôt qu'en bas à gauche.

On a déjà décrit la Règle 8 qui permet de trouver la source la plus proche d'un objet considéré, parmi un ensemble de sources candidates.

- 25 La Règle 9 qui suit permet de déterminer quelles sont les sources candidates d'un objet considéré. Les différents attributs d'un objet (contenu, position¹⁵³, transformation de transclusion, etc) sont obtenus indépendamment, d'après cette règle, et il se pourrait, par exemple, qu'un objet obtienne sa position depuis un objet, et son contenu depuis un autre.

- 30 Notons tout d'abord que chaque source candidate de l'objet considéré doit être une source (à l'amont) de l'objet considéré, et donc qui lui correspond par rapport à une transclusion¹⁵⁴ ou une chaîne de transclusions¹⁵⁵. Ceci est un critère supplémentaire qui s'ajoute à chacun des critères énoncés ci-dessous.

Règle 9

Est une source candidate pour un attribut d'un objet considéré :

1. toute source dudit objet considéré, dont l'attribut en question a été modifié ;
- 35 2. toute source –ou objet contenu dans une source– d'une transclusion fournissant sa transformation localement¹⁵⁶ et dont le résultat est –respectivement englobe¹⁵⁷– ledit objet considéré ;
3. tout résultat¹⁵⁸ –ou objet contenu dans un résultat qui n'englobe pas ledit objet considéré– d'une transclusion fournissant sa transformation localement ;

¹⁵³ (sauf lorsque l'on adopte l'approche des « Créneaux » déjà décrite selon laquelle un changement de position est vu comme une modification du contenu du parent)

¹⁵⁴ (dudit objet ou d'un ancêtre)

¹⁵⁵ (cette dernière va s'appliquer pour la règle 9.1 et va être pris en compte dans le cadre de la règle 9.4)

¹⁵⁶ On dit qu'une transclusion fournit sa transformation localement si elle a été créée explicitement ou si sa transformation a été modifiée.

¹⁵⁷ Un objet A englobe un objet B si A est le parent ou un ancêtre de B.

¹⁵⁸ (qui n'est bien sûr pas ledit objet considéré ni un objet qui l'englobe)

4. tout résultat –ou objet contenu – résultat qui n’englobe pas ledit objet considéré– d’une transclusion dont la source est une source candidate pour ledit objet considéré¹⁵⁹.

5 Une des sources candidates pour le contenu ou la position, étant sélectionnée comme étant la plus proche en utilisant la règle 8, dans le cas où la transclusion qui l’amène ne fournit pas sa transformation localement (cas des transclusions implicites) il faut aussi trouver la source pour la transformation à y appliquer. Ce sera celle qui est source candidate la plus proche, cette dernière étant déterminée selon les mêmes règles (9 et 8).

La prise en compte ou non d’un attribut dépend du type de la transformation.

10 **Exemples de transformations**

Exemples introductifs : remplacement et anti-doublons

L’exemple illustré à la figure 113 utilise deux fonctions différentes, « r » qui effectue le remplacement de tous les 1 par des 2, et « u » qui supprime tous les doublons dans le texte des objets (mais pas entre les objets).

- 15 Le rectangle du haut représente un objet contenant une transclusion interne dont la transformation est « r ». Cet objet est la source d’une deuxième transclusion dont le résultat est tout en bas. La transformation associée à cette deuxième transclusion est la fonction « u ».

20 Dans les figures 113 et 114, les lignes en traitillés indiquent les emplacements où une transformation est appliquée ; les flèches en pointillés indiquent les flux d’information et les flèches pleines indiquent les transclusions ; le contenu des rectangles en bas des figures indique ce qui est montré à l’utilisateur ; la règle utilisée pour déterminer la source candidate qui a été sélectionnée est indiquée entre crochets.

25 Dans cet exemple, aucune modification locale n’a été effectuée dans un résultat de transclusion, et la source utilisée pour chaque objet visible dans la figure est l’objet a dans le coin supérieur gauche.

30 Pour l’objet 2:1:a le flux d’information passe par 1:a et non pas par 2:a. En effet 1:a est une source candidate (selon la règle 9.2) ce qui n’est pas le cas de 2:a (la règle 9.3 ne s’applique pas car 2:a est dans le résultat d’une transclusion qui fournit bien sa transformation localement mais ce résultat englobe l’objet considéré 2:1:a). Ainsi dans la figure il n’y a pas de flèche partant de 2:a, qui n’est donc jamais soumis à la fonction de transformation.

Avec modifications locales

35 On voit que maintenant, suivant la règle 9.1 (une source modifiée est une source candidate) ainsi que la règle 8 (2:a est plus proche que 1:a), l’information pour l’objet 2:1:a vient directement de 2:a, et que ce dernier est d’abord soumis à la fonction « r » avant d’être propagé en 2:1:a. Et à l’inverse, la fonction « u » n’est plus utilisée de l’objet 1:a à l’objet 2:1:a.

Filtres de sélection par rapport à un critère

Comme autre exemple de fonction pouvant être utilisée avec une transclusion, on peut effectuer un filtrage des objets, ne laissant passer que ceux qui répondent à un certain critère.

- 40 Le filtre étant effectué sur les objets de la source il reste possible pour l’utilisateur d’ajouter n’importe quel objet dans le résultat de la transclusion, même s’il ne répond pas au critère du filtre. (A noter cependant que si l’objet est accepté à l’amont il repasserait de l’autre côté du filtre et deviendrait ainsi invisible à l’utilisateur).

¹⁵⁹ Cette règle s’applique donc récursivement. Elle a comme effet que les sources candidates ont toujours distantes d’une seule relation de transclusion.

Remplacement de transformatio

Les transformations à faire appliquer par une transclusion peuvent être modifiées, voire transformées par des transformations de transclusions.

- 5 Les figures 114 à 118 montrent ce qui se passe quand des transformations sont modifiées. Ici encore, les flèches pointillées indiquent les transferts d'informations et sont annotées avec la fonction qui est appliquée au passage. Les rectangles représentent des objets et les informations soulignées indiquent des modifications locales (les valeurs non soulignées étant obtenues de façon implicite). Les objets sont nommés par les valeurs qu'ils sont censés fournir (par exemple pour la figure ci-après l'objet le plus à droite en bas sera appelé $k(g(f(1)))$). Entre crochets est indiquée la règle appliquée pour sélectionner la source candidate.

- 10 Voici tout d'abord, dans la figure ci-dessous les valeurs obtenues lorsque aucune modification n'est effectuée dans un résultat de transclusion. On voit dans la figure que la source candidate la plus proche de l'objet considéré « $k(g(f(1)))$ » est « $g(f(1))$ ». En effet, « $g(f(1))$ » est bien une source pour l'objet considéré et il l'est pour la transclusion qui fournit sa transformation (k) localement, donc la règle 9.2 s'applique, alors que les objets « $k(g(1))$ » et « $k(f(1))$ », qui sont aussi des sources, ne sont pas source de transclusion fournissant une transformation localement (donc la règle 9.2 ne s'applique pas), et ne sont pas non plus un résultat d'une transclusion qui fournit sa transformation localement, ce résultat n'englobant pas l'objet « $k(g(f(1)))$ » considéré (donc la règle 9.3 ne s'applique pas non plus).¹⁶⁰
- 15 Dans la figure 115 on a remplacé $k(1)$ par la valeur 4. On va maintenant décrire l'application des règles 8 et 9 après ce remplacement.¹⁶¹

- 20 En application des règles 9.1 et 8, 4 est source candidate la plus proche pour « $k(f(4))$ » et « $k(g(4))$ ». ($k(f)$ est la transformation f transformée le cas échéant¹⁶² par k ; $k(g)$ est la transformation g transformée le cas échéant par k .) Quant à l'objet le plus à droite en bas $k(g)(k(f(4)))$, la règle 9.4 lui permet de prendre $k(f(4))$ comme source candidate qui est plus proche que $g(f(1))$. A noter que $k(g(4))$ n'est pas source candidate car il est dans le résultat de la transclusion « $k(g)$ » qui englobe $k(g)(k(f(4)))$, et donc la règle 9.4 ne peut pas s'appliquer.

- 25 Dans la figure 116, trois modifications ont été effectuées sur la figure 114 : $g(1)$ a été remplacé par la valeur 2, la transformation $k(f)$ a été remplacée par h en haut à droite et en bas à gauche la transformation implicite a été remplacée par t .

30 Comme en haut à droite la transformation a été modifiée en h , pour l'objet considéré nommée $h(k(1))$ dans la figure, la source $k(1)$ de la transclusion h est considérée comme source candidate la plus proche¹⁶³ en vertu des règles 9.2 et 8.

¹⁶⁰ Enfin, selon la règle 8, $g(f(1))$ est plus proche que les autres sources candidates. L'utilisation de la règle 8 étant en général évidente, on ne la mentionnera pas systématiquement.

¹⁶¹ A noter ici qu'on peut aussi utiliser la méthode suivante :

Dans la figure précédente, f est un attribut de l'objet $f(1)$; k est un attribut des objets $k(1)$, $k(f(1))$, $k(g(1))$ et $k(g(f(1)))$; g est un attribut des objets $g(1)$ et $g(f(1))$.

Remplacer $k(1)$ par 4 signifie que l'attribut k est supprimé et le contenu modifié en 4. Alors, pour $k(f(1))$ la source la plus proche pour le contenu est maintenant 4 ; la source la plus proche pour k est 4 où elle est supprimée ; la source la plus proche pour f est toujours $f(1)$; donc $k(f(1))$ devient $f(4)$; pour $k(g(1))$ la source la plus proche pour le contenu est 4 ; la source la plus proche pour g est toujours $g(1)$; la source la plus proche pour k est 4 où elle est supprimée ; donc $k(g(1))$ devient $g(4)$; pour $k(g(f(1)))$ la source la plus proche pour le contenu est 4 ; la source la plus proche pour k est 4 où elle est supprimée ; la source la plus proche pour g est toujours $g(f(1))$; la source la plus proche pour f est toujours $g(f(1))$ dont la source est $f(1)$ donc la source la plus proche est $f(1)$; donc $k(g(f(1)))$ devient $g(f(4))$.

¹⁶² Cela dépend de la nature de la transformation k . A noter que ce cas de figure est rare : en général, les transformations ne transforment pas des transformations mais des contenus et/ou positions... Lorsque k ne transforme pas f ou g , on dira que k (dans $k(f)$ ou $k(g)$) est la fonction identité. Même raisonnement pour les autres attributs...

¹⁶³ Ainsi $h(k(1))$ prend sa source dans 1 via k et h (et non via f et k).

Pour l'objet considéré $k(g)(h(k(1)))$, h ⁸⁷ sont des sources candidates selon la règle 9.3, mais $h(k(1))$ est la plus proche (selon la règle 8).¹⁶⁵

La figure 117 est le même exemple, avec juste une modification en plus, $k(2)$ ayant été remplacé par la valeur 3.

- 5 L'objet tout en bas à droite est intéressant par le fait que la source candidate la plus proche est « 3 » pour le contenu (règles 9.1 et 8) et ensuite¹⁶⁶ la source candidate la plus proche est « h » pour la transformation et cette dernière est elle-même transformée par $k(g)$, donc la transformation à appliquer est $k(g)(h)$ ¹⁶⁷. On observe ainsi qu'on obtient la transformation à effectuer et la valeur sur laquelle effectuer cette transformation de deux endroits différents.¹⁶⁸
- 10 Les figures 118 et 119 illustrent davantage l'utilisation de ces mêmes règles dans le cas de transformations de transformations. On repart du dernier exemple et la source 1 est maintenant remplacée par deux sous-objets, « 5 » et « $s(5)$ », le deuxième étant le résultat de la transclusion du premier, transclusion à laquelle est associée la transformation s . On suppose qu'ici que l'on adopte l'approche des créneaux et que donc les deux nouveaux sous-objets ne
- 15 sont pas propagés vers l'objet qui contient 2 .

Dans la figure 119, le contenu de l'objet le plus à droite en haut est remplacé par la valeur 6. L'objet nommée « $h(k(s))(6)$ » est maintenant le résultat d'une transformation qui est elle-même le résultat de la transformation par h de la transformation par k de la transformation s .¹⁶⁹

Principes d'intégration à un tableur tel que Microsoft Excel

- 20 Voici comment le système de transclusions peut s'intégrer à Excel¹⁷⁰ :

Une extension d'Excel va offrir à l'utilisateur une fonction de création et gestion de transclusion. Dans la suite cette extension est parfois désignée « le système » ou « l'interface ».

- 25 Pour effectuer une transclusion, l'utilisateur sélectionne un ensemble de cellules (formant un « rectangle »¹⁷¹), en effectue un glisser-déposer et au moment du « déposer » choisit dans le menu contextuel l'option « transclusion »¹⁷² fournie par l'interface.

Lorsque cette opération est effectuée, le système crée un objet pour le rectangle de cellules, dont les objets enfants sont les cellules du rectangle. Dans le cas où certaines de ces cellules

¹⁶⁴ Pour la source $k(2)$ aucune règle ne s'applique puisque 1) cet objet n'est pas modifié, 2) n'est pas source d'une transclusion qui fournit sa transformation localement, 3) n'est pas contenu dans un résultat qui n'englobe pas l'objet noté $k(g)(h(k(1)))$ dans la figure...

¹⁶⁵ Ainsi, $k(g)(h(k(1)))$ prend sa source dans 1 via k , h et g (et non dans 2 via t et k).

¹⁶⁶ Comme la transclusion à partir de cette source « 3 » ne fournit pas sa transformation localement, on doit déterminer laquelle prendre.

¹⁶⁷ (où, comme déjà expliqué, $k(g)$ peut être ici une fonction de transformation de transformation ou la fonction identité)

¹⁶⁸ La figure 117 ainsi que les figures 118 et 119 montrent que la transformation h est individuellement transformée en une transformation $k(g)(h)$. Toutefois, comme déjà évoqué, h peut être mise en œuvre sous la forme d'un attribut du résultat de la transclusion à laquelle cette transformation est associée.

¹⁶⁹ Rappelons encore que chaque transformation ne va pas nécessairement transformer la transformation qui est à l'amont, elle peut en effet se contenter de transformer le contenu et/ou la position (ou même être la fonction identité c'est-à-dire ne rien transformer), mais on décrit ici le cas le plus général.

¹⁷⁰ Par Excel on entend ici n'importe quel tableur. Excel (Microsoft Excel) est une marque déposée de Microsoft.

¹⁷¹ On peut facilement mettre en œuvre les mêmes fonctionnalités pour un ensemble de cellules qui ne forment pas un rectangle, voir l'approche présentée plus bas.

¹⁷² Dans les applications il y a en général (notamment lorsque l'on utilise le bouton droit de la souris) les options « Déplacer ici », « Copier ici », « Créer ici un lien ». On propose d'ajouter l'option « Transclure ici ».

avaient déjà été groupées dans un objet
nouvel objet¹⁷³.

88

jet et non elles qui devient enfant du

On va maintenant présenter une mise en œuvre permettant d'associer des transformations aux transclusions et de les gérer.¹⁷⁴

- 5 Les transformations sont ici décrites sous forme de « formules Excel » à un paramètre.¹⁷⁵

Pour les traiter correctement le système doit pouvoir analyser des formules Excel et y repérer les dépendances. On dit que *a dépend de b* si la formule définissant *a* contient une référence à *b*.

L'approche est la suivante :

- 10 Supposons (comme le montre la figure 120) que l'utilisateur ait fait une transclusion 1 d'un objet *a* (le résultat étant l'objet 1:*a*) avec une fonction *g(x)* comme transformation. Disons que l'objet *a* contient une cellule A2.

- 15 Dans cet exemple, en suivant la règle 9.2 (comme la transclusion 1 fournit sa transformation *g(x)* localement), la formule inscrite dans la cellule correspondante B2 sera initialement =*g(A2)*.

Supposons que la cellule A2 soit définie par une formule *f* qui *dépend* d'une cellule A1 se trouvant également dans la source de la transclusion (*f* peut éventuellement dépendre d'autres cellules également).

- 20 Supposons maintenant que la cellule B1 qui correspond à A1 par rapport à la transclusion 1, soit modifiée par l'utilisateur, comme le montre la figure 121. Pour appliquer les règles 9.1 et 8 pour l'objet considéré B2, le système doit alors changer la valeur inscrite dans B2 pour faire maintenant référence à B1 (puisque B1 vient de devenir une source candidate selon cette règle 9.1 et que selon la règle 8 elle est plus proche que A2 ou A1). Pour ce faire, le système va simplement reproduire la formule « =*f(A1)* » de A2 en décalant les références qu'elle
25 contient (ceci revient à effectuer programmatiquement le copier-coller de A2 dans B2, ainsi au lieu de =*f(A1)* qui est la formule définissant la cellule A2, pour définir la cellule B2 Excel mettra =*f(B1)* dans cette dernière). On dit que la nouvelle formule qui définit B2 est la formule *f(A1)* *transposée*.

- 30 Ainsi l'image d'une formule à travers une transclusion est vue comme une transclusion *implicite*. Lorsqu'une source d'une telle transclusion implicite est modifiée pour la première fois, le système modifie le résultat de cette transclusion implicite automatiquement pour qu'il reflète ladite formule et qu'Excel puisse exécuter cette formule sur la source en question (en application des règles 9.1 et 8). Avantageusement on laisse ainsi Excel appliquer la transformation associée à cette transclusion implicite.

¹⁷³ Dans la mesure où un objet ne peut avoir plus qu'un parent, il faut refuser d'effectuer une transclusion qui aurait comme conséquence un chevauchement d'objets, c'est-à-dire si la source ou le résultat d'une transclusion intersecte la source ou le résultat d'une transclusion éventuellement différente, sans que l'une ne soit contenue dans l'autre. En variante, on peut considérer que les objets ayant plusieurs parents sont traités comme s'ils étaient des objets différents (avec des identifiants différents).

¹⁷⁴ La mise en œuvre dans Excel de la fonctionnalité de transclusion « sans transformation » peut être comme suit: lors de la création de la transclusion et à chaque fois qu'une cellule de la source change, le résultat est mis à jour automatiquement en exécutant la fonction copier-coller d'Excel, copiant les cellules de la source aux cellules du résultat. A noter que les cellules contenant une formule ne sont pas traitées différemment des autres, car la fonction copier-coller sait déjà les traiter correctement, décalant les références¹⁷⁴ si nécessaire.

¹⁷⁵ On peut aussi avoir comme transclusions des formules à plusieurs paramètres, l'approche est alors de considérer un objet virtuel, qui englobe toutes les cellules représentées par ces paramètres, comme étant la source de la transclusion. Mais une même cellule pourrait alors avoir plusieurs parents virtuels. On a déjà évoqué (juste un peu plus haut) l'approche consistant à traiter les objets ayant plusieurs parents (en les traitant comme s'ils étaient des objets différents tout en les faisant se comporter in fine en un même objet).

En contrepartie de cet avantage, dans le transclusion, le système doit surveiller les cellules en question (c'est-à-dire les sources de ladite transclusion implicite) et effectuer un changement de formule lorsqu'un changement de la règle appliquée (Règles 9.1 à 9.4) l'exige. En particulier si l'utilisateur annule sa modification de B1, la formule précédente pour B2

5 devra être remise en place.

Pour illustrer ce procédé de manière plus concrète, voici un autre exemple. Supposons qu'une cellule A3 contienne la valeur $=A2+2$, et qu'une transclusion de A2 et A3 soit effectuée, faisant correspondre A3 à B3. Si la transformation $(x+1)*C8$ est associée à cette transclusion, x étant le paramètre source, la transclusion inscrit $=(A3+1)*C8$ dans la cellule résultat B3, et

10 $=(A2+1)*C8$ à la position B2. Si ensuite l'utilisateur remplace la valeur de B2, par exemple pour écrire 10, la définition de B3 est automatiquement changée pour devenir $=B2+2$.

Ainsi, le système est conçu pour que le cas d'une cellule faisant référence à une autre se comporte de la même façon que si cette référence est créée à l'aide d'une transclusion, bien qu'il y ait quelques différences de traitement :

- 15 • Lorsque l'utilisateur inscrit dans une cellule une formule contenant une référence vers une autre, le système (c'est-à-dire l'extension que nous décrivons ici) n'intervient pas (et laisse faire Excel comme s'il n'y avait pas d'extension). Toutefois, comme décrit ci-dessus, le cas d'une transclusion dont la source contient une cellule comportant une
- 20 référence à une autre cellule de la source est traité spécialement (en analysant les références dans les formules) afin de reproduire le comportement des transclusions internes.
- Lorsque l'utilisateur transclut plusieurs cellules à la fois, il n'a besoin d'indiquer la transformation à effectuer qu'une seule fois. Par exemple lorsqu'il change cette transformation toutes les cellules résultat sont mises à jour.
- 25 • Une différence conceptuelle est qu'une formule peut contenir des références à plusieurs cellules, contrairement à une transclusion qui n'a qu'une source.¹⁷⁶

Ainsi le procédé de propagation des manipulations le long des transclusions dans Excel est le suivant :

Règle 10 :

- 30 1. Soit un objet 1:a, image d'un objet a par une transclusion 1. Soit $\{b\}$ l'ensemble des références se trouvant dans la formule donnant la valeur de l'objet a¹⁷⁷. Pour l'objet 1:a considéré (« considéré » au sens de la règle 9), voir tous les $\{b\}$, ainsi que leurs correspondants 1: $\{b\}$ (lorsqu'ils existent), comme des sources. Appliquer la Règle 9 pour déterminer parmi ces sources lesquelles sont des sources candidates.
- 35 2. Étant donné que (selon la Règle 8) pour n'importe quel b pris dans $\{b\}$, 1:b sera forcément plus proche que a et que b, si n'importe quel 1:b de 1: $\{b\}$ est une source candidate, alors l'ensemble 1: $\{b\}$ est considéré comme source candidate. La formule à indiquer dans 1:a est alors la transposition de a, contenant les références vers 1: $\{b\}$. Dans le cas contraire, c'est f(a) qui doit être placée dans 1:a, f() étant la transformation
- 40 associée à la transclusion 1.

A noter que dans le cas de cellules de $\{b\}$ ne se trouvant pas dans la source de la transclusion, il peut être préférable de ne pas transposer leurs références, les considérant comme absolues. Ceci reproduirait donc le comportement en cas de transclusions indépendamment de Excel.

¹⁷⁶ Comme déjà dit, le cas de transclusion dont la formule contient des références à plusieurs sources peut être pris en compte par le système en assimilant ces sources à des descendants d'une source virtuelle unique.

¹⁷⁷ $\{b\}$ est un ensemble vide si a est défini par une constante.

Les exemples des figures 122 à 126 r lation des règles vues jusqu'ici. Ils illustrent comment les transclusion internes ou non et les références par formules interagissent. Les valeurs ou formules indiquées dans les cellules sont celles qui sont visibles à l'utilisateur, le calcul effectif des formules étant laissé à Excel. Les valeurs fournies par l'utilisateur sont soulignées, les autres étant obtenues implicitement par le système.

La figure 122 montre l'état des cellules en l'absence de modification dans les résultats de transclusion. Toutes les cellules dans un résultat de transclusion contiennent une référence à la cellule correspondante à la source, à laquelle est appliquée la transformation associée à la transclusion. La partie de la règle 9 qui justifie la formule dans chaque cellule est indiqué entre crochets.

Voici maintenant ce qui se passe si l'utilisateur effectue deux modifications, respectivement dans B1 et A11.

Comme A2 contient une référence à A1 (sans cependant être une transclusion) et que la correspondante de A1 (B1) est modifiée dans le résultat de la transclusion 1, suivant la règle 10, B2 (qui initialement contenait la valeur $=A2*2$) contient maintenant la transposition de A2.

A10 n'ayant pas été modifiée localement, B10 obtient toujours son contenu de B1 (source modifiée la plus proche, règle 9.1), transformé par $6+x$.

La source candidate la plus proche de B11 est A11. La formule de B11 est donc obtenue en appliquant la transformation $x*2$ à A11 (au lieu de la transformation $6+B2$).

Enfin, dans la figure 123, on reprend l'exemple initial (présentée dans l'avant-dernière figure) et on suppose qu'au lieu de $6+A1$, la cellule A10 contienne la constante 100.

En conséquence, suivant la règle 10, comme la source (A2) de A11 dépend de A1 et que le correspondant (A10) de A1 est maintenant modifié, A11 reçoit maintenant $=A10+2$, la transposition de A2 (en lieu et place de $6+A2$).

La source modifiée la plus proche de B10 est maintenant A10. La formule de B10 est donc obtenue en appliquant la transformation $x*2$ à A10. De même, la source candidate la plus proche de B11 est maintenant A11 (en vertu de la règle 9.4).

On voit donc que les références par formules se comportent bien comme des transclusions internes.

Objets multi-versions

La technologie des transformations associées aux transclusions peut être utilisée pour gérer des versions d'objets transformées en fonction des utilisateurs.

Rappelons qu'un ERC est vu comme un groupe d'utilisateurs (indépendant du concept correspondant dans le système d'exploitation). Ce concept est utilisé par le système pour décider quels objets rendre accessibles à quels utilisateurs.¹⁷⁸

Une ressource est toujours visionnée dans le cadre d'au moins un ERC (et ledit ERC peut être sélectionné en lecture seule¹⁷⁹).

Pour l'enregistrement de données on spécifie dans quel ou quels ERC ces données doivent être visibles (il s'agit d'ERC sélectionnés en écriture).

¹⁷⁸ Ce contrôle d'accès peut être placé au-dessus de ou après le contrôle effectué par le système d'exploitation.

¹⁷⁹ Pour un ERC auquel l'utilisateur a le droit d'accès en écriture, l'ERC peut être ouvert en lecture seule (afin de recevoir les contributions des autres utilisateurs de l'ERC en question mais ne rien contribuer soi-même dans cet ERC). L'interface permet de basculer facilement du mode lecture au mode lecture-écriture.

De préférence, il existe par défaut un I⁹¹ r, nommé E0 ou « ERC par défaut », auquel tout le monde a accès au moins en lecture, qui est utilisé pour produire des données sans restrictions.

5 Les *objets multi-versions* sont des objets qui sont présentés de façon différente selon l'ERC utilisé pour les obtenir.

Ils sont définis par un objet de base et, pour chaque ERC, une transformation à appliquer sur cet objet de base pour obtenir la version correspondant à l'ERC considéré (on peut aussi transformer des versions qui ont été eux-mêmes obtenues par transformation sur un objet de base pour obtenir encore d'autres versions).¹⁸⁰ Les ERC pour lesquels aucune version n'est
10 spécifiée n'ont simplement pas du tout accès à cet objet.

Lorsqu'une version d'un objet multi-versions (ou l'un de ses descendants dans la structure de la ressource) est utilisé comme source de transclusion, c'est une transclusion de l'objet multi-versions qui est effectuée.¹⁸¹

Exemple d'objet multi-versions

15 Le document « a » présenté à la figure 125 contient une liste d'articles sur des sujets variés.

On considère trois ERC : E1, E2 et E3.

En accédant au document par l'ERC E1 on en voit l'ensemble du contenu, alors qu'en accédant par l'un des deux autres seul un sous-ensemble (filtré par les transformations f_{E2} et f_{E3} par exemple en fonction du sujet des articles) est affiché.

20 Des modifications effectuées via E1 sont propagées aux deux autres versions (dans la mesure où les filtres les laissent passer). Une modification effectuée avec l'un des deux autres ERC est visible uniquement avec l'ERC en question¹⁸².

Supposons qu'un utilisateur visionne le document par l'ERC E2, choisisse deux articles a2 et a3, et en effectue une transclusion dans un nouveau document n.

25 Supposons que lors de la création de ce dernier il ait demandé à ce qu'il soit visible pour tout le monde (et non restreint à l'ERC E2). Pour ce faire il a sélectionné E0 en écriture.

La structure alors obtenue est montrée dans la figure 126. Les lignes en pointillés sont les transformations menant aux différentes versions, les cadres traitillés délimitent les objets multi-versions et pour chaque version l'ERC concerné est indiqué à gauche.

30 Interface utilisateur

L'interface permet à l'utilisateur d'ouvrir (sélectionner) un certain nombre d'ERC (parmi ceux où il a l'accès) en lecture seule ou aussi en écriture selon les droits de l'utilisateur. *Les ERC sont ordonnés par ordre de préférence de l'utilisateur et l'utilisateur peut modifier cet ordre.*

35 Pour un ERC auquel l'utilisateur a le droit d'accès en écriture, l'ERC peut être ouvert en lecture seule (afin de recevoir les contributions des autres utilisateurs de l'ERC en question mais ne rien contribuer soi-même dans cet ERC). L'interface permet de basculer facilement du mode lecture au mode lecture-écriture.

¹⁸⁰ Comme décrit plus loin, l'objet multi-version possède des enfants correspondant chacun à un ERC différent, reliés entre eux par des transclusions portant des transformations. L'enfant correspondant à l'ERC courant est automatiquement sélectionné en lieu et place de l'objet multi-version.

¹⁸¹ Autrement dit, lorsqu'une transclusion d'un descendant d'une version est effectuée, la véritable source de la transclusion est l'ensemble de l'objet multi-version, et la transclusion est munie d'une transformation effectuant une sélection de l'objet en question.

¹⁸² (mais peut être contribué à l'amont, c'est-à-dire à E1)

Lors de l'affichage d'un objet multi-versur, les versions pour lesquelles l'ERC est ouvert sont accessibles via l'interface, et celle en premier dans l'ordre de préférence est affichée par défaut.

- 5 Si aucun des ERC pour lesquels il existe une version n'est ouvert, rien n'est affiché, comme si l'objet n'existait pas.

- 10 Le système fourni à l'utilisateur deux moyens pour changer les versions affichées. Tout d'abord, changer l'ordre de préférence dans la liste des ERC ouverts affecte immédiatement les sélections automatiques. D'autre part, individuellement pour chaque objet multi-version présent à l'écran l'utilisateur a moyen de changer la version affichée, par exemple à l'aide d'onglets, ou d'un bouton s'affichant lorsque le curseur passe sur l'objet ou lorsque ce dernier est sélectionné.

La figure 127 montre comment se présente l'interface utilisateur par défaut lorsque l'utilisateur visionne le document « n » de l'exemple précédent, en ayant les ERC E10, E3, E2, E1, E0 sélectionnés comme ouverts, dans cet ordre de préférence.

- 15 Comme il n'y a des versions de a2 que dans E2 et E1, et E2 est préféré à E1, c'est la version de E2 qui est affichée, alors que c'est la version de E3 qui est affichée pour l'article a3 puisqu'il y en a une dans E3 et que cet ERC est préféré à E2 et E1. Bien entendu les onglets (permettant pour chaque objet multi-versions de sélectionner la version pour un ERC donné) ne sont affichés que dans un mode activé (c'est-à-dire que le mode onglets est
- 20 activable/désactivable) par l'utilisateur.

Lorsque l'utilisateur sélectionne manuellement un onglet d'un objet multi-versions, le système se souvient de cette sélection et la restitue la fois suivante que l'utilisateur affiche la ressource.¹⁸³

On va maintenant décrire l'interface utilisateur pour les ERC sélectionnés en écriture.

- 25 Concernant la création d'objets, que ce soit une création ex nihilo ou un résultat de transclusion, l'interface peut être placée en deux modes, sélection manuelle d'ERC ou sélection automatique.

En mode automatique, les objets ajoutés sont placés dans le même ERC que leur parent.

- 30 En mode manuel, l'utilisateur choisit parmi les ERC ouverts en lecture, un ERC en écriture qui par défaut recevra les nouveaux objets. Si l'utilisateur a choisi l'ERC E1 pour les nouveaux objets et crée un objet dans la version E1 d'un objet, le nouvel objet est simplement ajouté directement. Si (comme dans la figure 128) l'ERC du contenant (c'est E2 dans la figure) est différent de l'ERC choisi E1, un nouvel objet multi-version est créé automatiquement, ne contenant initialement qu'une version pour E1.

- 35 Pour créer un objet multi-version, l'utilisateur commence par créer la version de base¹⁸⁴ de cet objet¹⁸⁵, puis après l'avoir sélectionnée demande à en créer une nouvelle version¹⁸⁶. Il peut alors choisir pour quel ERC il faut créer la nouvelle version (l'interface doit cependant empêcher de créer deux versions du même objet pour le même ERC).

- 40 L'interface offre également un outil de transfert (copie, déplacement, transclusion, etc.) entre versions d'un objet multi-version. Si comme dans la figure 128, la version E1 de l'objet a contient un objet b (et a est le premier ancêtre de b à être multi-version), l'utilisateur

¹⁸³ La configuration des versions sélectionnées (pour une ressource) peut être enregistrée (dans un serveur).

¹⁸⁴ Dans le cas où la version de base est créée dans un ERC différent de celui du parent, celle-ci sera créée directement comme objet multi-versions.

¹⁸⁵ (ou choisit un objet existant qui n'est pas multi-version)

¹⁸⁶ Par exemple par le menu contextuel. Dans le cas où l'objet est déjà multi-version, cet outil peut être présenté au niveau des onglets, sous forme d'un bouton « ajouter un onglet »

sélectionne b puis active (par exemple via un bouton contextuel) la fonction de transfert qu'il souhaite utiliser. Le système lui permet alors de choisir dans la liste des versions de a, présentée comme une liste d'ERC, celle qui sert de destination de l'opération. Au cas où il existait déjà un objet correspondant¹⁸⁷, le système affiche les deux objets et propose de remplacer l'objet déjà existant.

A noter qu'il est possible d'ouvrir plusieurs « fenêtres » simultanément (ce qui est schématisé dans la figure 129), ce qui est par exemple utile si l'utilisateur souhaite voir simultanément plusieurs versions du même objet et effectuer des opérations de transfert plus complexes entre descendants de ces deux versions, par exemple effectuer des transferts entre deux points séparés par plus d'un objet multi-versions.

Mise en œuvre à l'aide de transclusions

On utilise le procédé de transclusion pour mettre en œuvre les objets multi-versions.

Comme déjà dit les différentes versions d'un objet sont obtenues de l'objet de base à l'aide de transformations. Pour obtenir ce résultat il suffit donc que toutes les versions soient des transclusions de cet objet de base, et que les transformations désirées soient associées aux transclusions. Cela nous donne la structure présentée dans la figure 130.

L'identifiant a* est donné à l'objet multi-versions. Maintenant, lorsque l'utilisateur demande à voir l'objet a*, c'est un parmi a, 2:a et 3:a qui lui est spontanément montré, en fonction de l'ERC utilisé. Ce remplacement s'effectue récursivement, dans le cas où la version affichée elle-même contient d'autres objets multi-versions.

Voici maintenant comment fonctionne l'exemple de transclusion vu plus haut.

La partie gauche de la figure 131 contient la structure de a* déjà vue plus haut.

Lorsque l'utilisateur effectue une transclusion d'un quelconque de ces objets, le système effectue en réalité une transclusion de a*. Le schéma suivant montre la transclusion (10) de 2:a2 (c'est-à-dire de a2 vu à travers l'ERC E2) et la transclusion (11) de 2:a3.

On voit que les objets multi-versions des résultats des transclusions sont des transclusions de a* lui-même. La fonction associée à ces transclusion est s(x), où x est l'identifiant de l'objet transclu par l'utilisateur, ce qui signifie que dans chaque version il faut uniquement garder cet objet x et ses différentes versions (avec leurs descendants).

L'objet multi-versions résultat de la transclusion 10 ne contient pas de version pour E3 car 3:a2 avait été filtré par f_{E3}.

Si l'utilisateur avait effectué par exemple des copies¹⁸⁸ plutôt que des transclusions, la structure représentée à droite aurait été reconstruite complètement plutôt que d'être implicitement décrite par des relations de transclusion.

Objets multi-versions imbriqués

Il peut arriver qu'il soit nécessaire de placer un objet multi-versions dans une version particulière d'un objet multi-versions.

Par exemple dans le scénario ci-dessus, si l'utilisateur ayant construit la page n décide de placer une annotation personnelle à l'intérieur de 10:2:a2, s'il souhaite être le seul à pouvoir la lire il faut qu'elle soit dans un ERC autre que E2.

¹⁸⁷ Plusieurs critères peuvent être utilisés pour établir cette correspondance. Comme expliqué plus loin, les différentes versions étant obtenues par transclusion, examiner les identifiants des objets est suffisant. On peut en plus traiter le cas où cet objet avait déjà été transclu ou copié explicitement, en vérifiant s'il existe déjà une transclusion ou copie de b dans la version choisie.

¹⁸⁸ (ou déplacements)

Supposons que E10 soit l'ERC particu⁹⁴ lisateur, l'annotation doit être visible seulement dans cet ERC. Pour ce faire, l'utilisateur crée un objet multi-version avec une seule version pour l'ERC E10, contenant son annotation.

5 Représenté sous forme d'arbre, le document n aurait alors une structure comme dans la figure 132.

Approches de mise en œuvre du contrôle d'accès

Pour être suffisamment général on considère que la structure des transclusions est stockée séparément des contenus, la première étant stockée dans des serveurs collaboratifs et les seconds dans des serveurs de contenus¹⁸⁹ (comme déjà décrit).

10 Normalement, chacun de ces serveurs effectue un contrôle d'accès et donc l'utilisateur s'identifie¹⁹⁰ avant de pouvoir lire ou modifier une quelconque de ces données. Pour chaque utilisateur la liste des ERC auxquels il appartient, accompagnée pour chacun du niveau d'accès (lecture ou écriture), est stockée dans un serveur destiné à cet effet.

15 Lorsque des utilisateurs échangent des ressources, une approche consiste à ce qu'ils échangent en réalité des références à ces dernières, permettant ainsi à plusieurs utilisateurs de collaborer sur une même ressource en n'en voyant que les versions auxquelles ils ont accès.

20 Une autre méthode consiste à chiffrer les objets avec des clés correspondant aux ERC autorisés à les voir. L'utilisateur possède une copie de la clé privée pour les ERC auxquels il appartient. Lorsqu'un utilisateur veut lire la ressource il déchiffre les parties concernées à l'aide de la clé appropriée, et lorsqu'il veut la modifier il signe ses modifications à l'aide de la même clé.

Avantageusement, la méthode utilisant du chiffrement permet à l'utilisateur d'avoir toutes les données localement, même celles auxquelles il n'a pas accès, et de les échanger ou copier par les moyens traditionnels.

25 Clés pour ERC en lecture et en écriture

Si l'on souhaite distinguer à ce niveau les droits d'accès en lecture et en écriture on peut utiliser des paires de clés différentes pour ces deux types d'accès. Ainsi une partie d'une ressource chiffrée pour un certain ERC est chiffrée avec sa clé de lecture « read » et signée avec sa clé d'écriture « write ». Ainsi, un utilisateur ayant la clé privée de lecture³⁰ read_{prv}+read_{pub} et seulement la clé publique d'écriture write_{pub} peut la lire et vérifier que l'auteur de la ressource avait bien la clé privée d'écriture mais ne pourra pas la modifier. Le schéma suivant illustre les transformations effectuées pour effectuer un accès en lecture ou en écriture au document chiffré. Voir la figure 133.

Serveur de chiffrement/déchiffrement

35 Un désavantage de la méthode basée sur le chiffrement est qu'il est difficile de retirer un utilisateur d'un ERC, car le système ne peut pas forcer l'utilisateur à supprimer sa clé. Il peut donc être nécessaire de changer la clé de l'ERC, mais les versions précédemment chiffrées devront alors être traduites pour la nouvelle clé.

40 Un autre problème est que le risque de divulgation d'information est plus grand, car si un utilisateur divulgue la clé d'un ERC à une autre personne cette dernière pourra déchiffrer les ressources sans que le système s'en aperçoive. Par exemple si l'utilisateur U1 appartient aux

¹⁸⁹ Bien entendu, ils peuvent aussi fonctionner comme un seul serveur ou même être mis en œuvre sur l'ordinateur de l'utilisateur.

¹⁹⁰ Un serveur d'authentification pourrait ici être utilisé pour éviter à l'utilisateur d'avoir à s'authentifier séparément pour chaque serveur.

ERC E1 et E2 et que l'utilisateur U2 n'a pas accès à E1, si U1 donne la clé d'E2 à U2, ce dernier pourra, tout en s'authentifiant normalement en tant que U2, accéder aux parties d'un document chiffrées pour E2, qui auraient dû lui être inaccessibles, et le serveur n'y verra rien puisqu'il s'est authentifié en tant qu'utilisateur ayant accès à E1 et qu'il peut donc accéder à ce document. Alors qu'en conservant tout sur le serveur le seul moyen à U2 d'obtenir des données d'E2 aurait été d'utiliser l'identité d'U1.

On peut donc considérer une méthode intermédiaire, combinant les avantages des deux approches :

Les utilisateurs ne possèdent pas les clés privées des ERC, qui sont plutôt placées dans un serveur de chiffrement/déchiffrement (qui peut être combiné au serveur d'authentification). Ce serveur stocke les clés privées de tous les ERC, ainsi que, pour chaque utilisateur, la liste des ERC auxquels il appartient (avec les niveaux d'accès).

Un utilisateur peut ensuite (une fois avoir prouvé son identité) soumettre au serveur de déchiffrement n'importe quel objet chiffré, et ce dernier en retourne la ou les versions auxquelles l'utilisateur a accès. Le même serveur peut également offrir l'opération inverse et signer numériquement les modifications apportées à l'objet, pour les utilisateurs autorisés.

Les utilisateurs peuvent donc toujours stocker les ressources localement et les échanger par les moyens traditionnels (restant ainsi indépendants des serveurs de contenus et des serveurs collaboratifs).

Le problème de révocation des clés ne se pose alors plus, puisque dès qu'un utilisateur est retiré d'un ERC¹⁹¹ le serveur de déchiffrement cesse d'accepter les requêtes de déchiffrement pour cet ERC.¹⁹²

Si cette méthode est adoptée, une seule paire de clé par ERC suffit, tant pour les droits en lecture que les droits en écriture, comme le serveur de déchiffrement vérifie les droits d'accès avant d'effectuer toute opération.

Cependant, cette méthode offre les désavantages liés aux échanges de ressources entre le poste client et le serveur de chiffrement/déchiffrement.

Stockage de clés en association avec les contenus

Le procédé de chiffrement/déchiffrement au niveau du poste client offre les avantages de :

- la décentralisation des traitements (éviter le goulet d'étranglement au niveau du serveur),
- de ne pas avoir à utiliser un serveur distant de chiffrement/déchiffrement de ressources (et notamment économiser ainsi de la bande passante)
- et surtout de pouvoir communiquer à des tiers des ressources chiffrées comportant toutes les versions existantes, y compris celles auxquelles l'utilisateur (celui qui communique) n'a lui-même pas le droit d'accès¹⁹³ mais qui seraient accessibles par un destinataire.

Mais un tel procédé peut présenter l'inconvénient déjà évoqué du risque de divulgation de clés à des tiers qui n'en ont pas (ou plus) droit et de leur permettre ainsi des accès non autorisés de manière permanente.

¹⁹¹ (pour un niveau d'accès spécifié)

¹⁹² (L'utilisateur pourra bien sûr avoir conservé des versions en clair d'anciennes ressources, mais ne pourra pas accéder aux ressources produites après la révocation).

¹⁹³ (en lecture ou écriture)

On peut limiter l'impact de telles divulgations¹⁹⁴ en ayant une clé spécifique pour chaque mise à jour (de chaque version) de contenu. Pour ce faire on utilise un serveur intermédiaire avec lequel l'utilisateur communique des clés¹⁹⁵ (plutôt que des contenus). On va maintenant décrire cette approche.

- 5 Lorsque l'utilisateur¹⁹⁶ chiffre un contenu pour l'enregistrer, il associe à ce contenu les paires de clés respectives des versions des objets multi-versions (par ERC¹⁹⁷) qu'il a générées, et cette association est stockée dans un serveur de clés¹⁹⁸.

10 Quand un utilisateur demande¹⁹⁹ d'accéder audit contenu (après s'être authentifié), les clés *correspondant à ses droits* lui sont transmises par le serveur de clés, en parallèle au contenu (chiffré) qui lui est fourni. L'utilisateur peut ainsi déchiffrer les versions du contenu auxquelles il a droit. (L'architecture de mise œuvre sous-jacente peut en fait faire intervenir d'autres serveurs intermédiaires tel que décrit plus loin dans la section présentant l'architecture globale).

15 Avantageusement, l'utilisateur peut transmettre le contenu chiffré, avec (toute les clés ou partie) des clés qu'il a reçu du serveur de clés, à un autre utilisateur par les moyens traditionnels (par exemple par courrier électronique), sans faire intervenir un serveur de contenu. L'autre utilisateur peut alors demander au serveur de clés les clés supplémentaires qu'il souhaite et auxquelles il a droit et le cas échéant avoir alors accès à des versions d'objets multi-versions auxquels le premier utilisateur n'avait pas lui-même accès.

20 Chiffrement des objets multi-versions

Lorsqu'une ressource contient des parties ayant des droits différents, voici comment la ressource chiffrée est structurée.

25 On appelle partie connexe d'une ressource un ensemble d'objets tel que pour toute paire d'objets de l'ensemble, lesdits objets sont reliés l'un à l'autre par un chemin formé de relations parent-enfant entièrement contenu dans l'ensemble.

Si tous les objets d'une partie connexe sont dans le même ERC, on parle de partie homogène (du point de vue des droits). On parle de partie homogène maximale si elle n'est pas contenue dans une partie homogène plus grande.

30 On considère l'ensemble des parties homogènes maximales, et à chacune on associe l'emplacement dans la ressource, typiquement sous forme de l'identifiant du parent de la

¹⁹⁴ On peut renouveler les clés pour limiter l'impact des divulgations : (1) renouveler les clés chaque fois qu'on retire à un utilisateur le droit d'accès (ou suite à une décision d'un administrateur qui juge qu'une clé a été divulguée) ; (2) renouveler les clés périodiquement, par exemple une fois par jour ; (3) renouveler les clés automatiquement à chaque création ou mise à jour d'un contenu (ledit contenu pouvant être une ressource, une partie de ressource, ou encore un simple objet) pour l'ERC en question. C'est l'approche (3) que l'on va maintenant décrire en détail. L'approche (1) peut être vue comme une optimisation de l'approche (3).

¹⁹⁵ Plus précisément l'utilisateur communique au serveur de clés des «associations (« clé-contenu-ERC ») entre notamment des clés et des identifiants de contenus que les clés permettent de déchiffrer : le serveur de clés sait quelle version de contenu peut être déchiffrée par quelle clé et donne cette information à qui en a droit.

¹⁹⁶ Par « l'utilisateur » on entend bien « (l'extension de) l'outil de manipulation d'informations que cet utilisateur utilise sur le poste client ».

¹⁹⁷ De préférence, le contenu en question est partitionné en différentes parties (chaque partie étant homogène du point de vue des droits d'accès). À chaque partie est associée une paire de clés. Ces clés sont stockées dans un serveur de clés ou avec le contenu, dans le serveur de contenu. Lorsqu'un utilisateur télécharge un contenu il reçoit le contenu complet, avec le sous-ensemble des clés correspondant à ses droits.

¹⁹⁸ Le serveur de contenus peut lui-même jouer le rôle de serveur de clés, dans ce cas on n'a pas besoin d'un serveur de clés séparé. L'avantage est alors que les clés sont stockés directement avec les contenus. Le désavantage est que l'on doit faire confiance aux serveurs de contenus...

¹⁹⁹ Les clés sont demandés au serveur de clés (en indiquant l'identifiant du contenu désiré ainsi que les ERC utilisés) en parallèle avec la demande du contenu lui-même à un serveur de contenu (ou en ouvrant le contenu en local dans l'ordinateur).

racine de cette partie (s'il existe) et du pr
partie on entend partie homogène maximale.

la racine (s'il existe). Dans la suite, par

Ensuite chaque partie est ainsi chiffrée et signée avec des clés pour l'ERC correspondant, et les données chiffrées sont ensuite placées en séquence, pour former la ressource chiffrée.

- 5 L'exemple de la figure 134 reprend la figure 132. Les lignes sont interrompues à l'emplacement de changement d'ERC. On suppose ici que les deux ERC multi-versions sont dans le même ERC que la racine de la ressource, alors que l'objet multi-version ayant servi à ajouter une annotation (dans E10) est dans le même ERC que son unique version. La raison pourrait être que le propriétaire de la ressource « n » a créé l'objet multi-version a* dans l'ERC E0, afin d'être sûr qu'il soit accessible à tout le monde, puis y a ajouté des versions pour les ERC E1, E2 et E3. En revanche lorsque l'utilisateur a ajouté l'annotation b, n'ayant pas accès en écriture dans l'ERC E2 (ou l'ayant désactivé), tant l'objet multi-versions b* que l'annotation elle-même sont dans le même ERC (et donc dans la même partie homogène maximale).

- 15 La structure chiffrée de la ressource sera donc la celle de la figure 135.

Chaque partie (représentée par un rectangle portant un nom d'ERC en étiquette) est chiffrée et signée avec une paire de clés correspondant à cet ERC.

- 20 Afin de permettre à des utilisateurs de copier des objets multi-versions d'une ressource à une autre sans risquer des conflits d'identifiants on peut partitionner la ressource en plusieurs « espaces d'identifiants ».

Chaque partie se trouve dans exactement un espace d'identifiants²⁰⁰, et les références²⁰¹ à d'autres objets, quand elles ne précisent pas d'espace d'identifiants, n'en franchissent pas les limites. Il est possible de franchir ces limites (l'exemple suivant indiquera comment) en nommant explicitement un autre espace d'identifiant de la ressource.

- 25 Ceci permet à un utilisateur de copier un objet multi-versions dans une autre ressource, même si l'utilisateur n'a accès en écriture ni au parent dans la ressource cible, ni aux versions de cet objet multi-versions (voir la figure 136). Il crée simplement un nouvel espace d'identifiants dans la ressource cible, copie toutes les versions souhaitées dans cet espace, puis y ajoute (toujours dans le même espace d'identifiants) un nouvel objet multi-versions du même identifiant que dans la ressource source. Pour placer ce nouvel objet multi-versions dans la structure de la ressource cible, il indique le nom de l'espace d'identifiants contenant le parent, en plus de l'identifiant de ce dernier.

- 30 Dans l'exemple de la figure 136, l'utilisateur souhaite copier les versions E1 et E2 de l'objet multi-versions 2*, depuis la ressource de gauche, sous l'objet 3 de la ressource de droite. Un nouvel espace d'identifiants IS2 est donc créé dans la ressource cible (et l'espace existant est appelé IS1), les deux parties contenant 3 et 11:3 y sont copiées sans qu'il soit nécessaire de les changer²⁰², et une nouvelle partie est placée dans cet espace, servant de connecteur entre ces versions et l'objet 3 de l'espace IS1.

- 40 Cette partie contient donc un objet (multi-versions) unique, qui spécifie comme parent l'objet 3 de l'espace d'identifiants IS1. Le double point indique qu'il faut tout d'abord sortir de l'espace d'identifiants IS2 avant de pouvoir entrer dans l'espace IS1 (à la manière d'une hiérarchie de répertoires).

²⁰⁰ (noté « IS » comme « Identifier Space » en Anglais)

²⁰¹ On utilise notamment les références pour indiquer la source d'une transclusion, ainsi que pour indiquer le parent et le prédécesseur de la racine d'une partie, comme sera expliqué plus loin.

²⁰² Elles n'ont donc pas besoin d'être déchiffrées.

Dans le cas où les parties à copier couvrent des espaces d'identifiants, il est nécessaire, dans la ressource cible, d'utiliser des espaces d'identifiants imbriqués, c'est-à-dire de créer un nouvel espace d'identifiant ne contenant aucune partie mais contenant les mêmes espaces que ceux utilisés par les parties du côté de la source (voir figure 137).

5 Possibilités de transfert

Voici les différentes opérations possibles dans le cadre des mécanismes décrits ci-dessus.

- Ajouter des objets (créés ou transclus, multi-versions ou non) dans une ressource (même à l'intérieur d'une partie qui n'est pas accessible en écriture, mais en lecture seule).
- 10 • Copier un objet multi-versions d'une ressource à une autre (ou dans la même)
- Copier une ou plusieurs versions d'un objet multi-versions soit dans une ressource, soit dans un nouvel objet multi-versions, soit dans un objet multi-versions existant.
- Supprimer une ou plusieurs versions d'un objet multi-versions.

15 On peut considérer deux variantes de la structure d'une ressource chiffrée, selon si les parties indiquent leur *position* (i.e. parent et prédécesseur) en clair (i.e. seulement de façon signée) ou si cette position est chiffrée.

20 La première variante permet, lors de la manipulation (copie, déplacement ou suppression) de versions, d'agir sur des versions non accessibles en écriture (car la position indique entre autres dans quel objet multi-versions ces parties se trouvent). En particulier cette variante permet, lors de la copie d'un objet multi-versions, que toutes les versions soient copiées, et non seulement celles accessibles en lecture (de manière à ce qu'un utilisateur ayant accès en lecture aux autres versions puisse les voir dans la ressource cible).

25 La seconde variante présente l'intérêt que si l'on n'a pas accès en lecture à des parties (telles que des annotations), on ne peut pas non plus savoir à quel endroit elles se trouvent dans la ressource. Cette variante est donc intéressante dans le cas d'annotations placées par un utilisateur.

Il est possible d'avoir dans une même ressource, et même dans un objet multi-versions des objets adoptant la première approche et d'autres adoptant la seconde.

30 A noter encore qu'il n'est pas possible (pour un utilisateur n'ayant pas accès en écriture) d'effectuer ces manipulations à un degré plus fin que celui des parties (par exemple copier seulement un objet d'une version d'un objet multi-versions), car cela impliquerait de modifier le contenu d'une partie, ce qui n'est pas possible même après l'opération de copie.

35 A noter que l'opération de transclusion d'un objet multi-versions ou d'un de ses descendants n'est pas concernée par ces considérations, par le fait que l'objet résultat est construit de façon implicite, et contient la référence à la source de la transclusion (contrairement à la copie qui ne garde pas cette référence, et qui ne suit pas les modifications à la source). Par conséquent, quelle que soit la variante utilisée la transclusion se comportera de la même façon, car un utilisateur qui peut accéder à un objet fonctionnant même selon la deuxième variante connaîtra sa position à la source de la transclusion et pourra en déduire (en appliquant l'algorithme de 40 transclusion) la position dans le résultat.

Exemple

La figure 138 montre le résultat d'une opération de copie d'une version d'un objet multi-versions. On voit qu'il a été nécessaire de restituer la source de la relation de transclusion. Les

figures en traitillés indiquent les parties c
accès en lecture.

des ERC auxquels l'utilisateur n'a pas

Cette opération est possible bien que l'utilisateur n'a accès en lecture qu'à la version pour E3 (dans le cadre de la première variante, où la position est publique).

- 5 Ensuite, l'annotation présente dans l'original a disparu, ceci se passe si on suppose que cette annotation utilisait la seconde variante, c'est-à-dire que sa position est chiffrée.

L'exemple de la figure 139 montre le résultat de la même opération, où l'utilisateur a demandé à copier l'ensemble de l'objet multi-versions plutôt que seule une version. On voit que la structure a été reproduite à l'identique dans la ressource cible, y compris les parties que
10 l'utilisateur ne peut lire (ici encore on suppose que ces parties utilisent la première variante).

A noter donc que dans ce cas de figure la source de la transclusion est transférée vers la partie correspondante (celle étant dans l'ERC E1) dans le ressource de droite.

Ici, pour la même raison que dans l'exemple précédent, l'annotation a disparu, en supposant comme précédemment que sa position est chiffrée.

15 **Objets de remplacement dans les objets multi-versions**

On va maintenant décrire comment exploiter le concept déjà décrit (voir la section Propagation des contributions dans les branches) d'objets de remplacement (et objets alternatifs) dans le cadre d'objets multi-versions.

- 20 Lorsqu'un utilisateur souhaite apporter une contribution dans une ressource, il peut créer un objet multi-versions (ou, le cas échéant, ajouter une version à un objet multi-versions déjà existant), dans le même ERC que le parent, ou dans le même ERC qu'une version déjà existante, mais *associée à son identité en tant qu'utilisateur*.

- 25 Du point de vue de l'interface utilisateur, comme le montre la figure 140, cela signifie que, en présence de contributions, au lieu qu'un objet multi-versions ait un ensemble de versions indiqué par un ensemble d'ERC, on aura en plus des versions étiquetées par des paires ERC/utilisateur. On peut donc avoir une situation mixte où il existe pour un ERC une version mise à jour collectivement (E1), ainsi que des versions individuelles de contributions (E1/U1 ; E1/U2) pour le même ERC.

- 30 Cette association peut se faire en attribuant à chaque utilisateur une paire de clés, et en faisant signer à chaque utilisateur les versions d'objets multi-versions qu'il produit.

- 35 Les modifications apportées dans sa version contributive seront donc propagées sous forme d'objets de remplacement dans la ou les autres versions du même objet et du même ERC, créées par d'autres utilisateurs. Plusieurs utilisateurs peuvent ainsi travailler ensemble sur une ressource, et se séparer en différentes branches en cas de refus (formant ainsi plusieurs groupes de travail pour un même objet, dans le même objet multi-versions).

- 40 Comme déjà dit, les différentes versions sont gérées sous forme de transclusions contenues dans l'objet multi-versions, ce qui permet aux objets de remplacement de circuler entre objets correspondant à la même source créée et se trouvant dans les différentes versions d'un objet multi-versions. Lorsqu'un utilisateur ajoute une version (étiquetée ERC/utilisateur) il peut choisir à partir de laquelle travailler, et dans la structure de l'objet multi-versions la version ajoutée par l'utilisateur est une transclusion de la version de base choisie.

- 45 La figure 141 montre la structure des objets dans le cas où une modification faite dans un résultat de transclusion (1:a) qui a été présentée comme objet de remplacement à sa source (a) a été refusée par cette dernière, résultant alors en un objet alternatif (2:1:a) dans le même ERC que celui de son contributeur et de manière étiquetée avec son identité d'utilisateur (E2/U1).

L'objet alternatif (2:1:a) est un résultat d à partir de la contribution refusée, pour que si cette dernière évolue, l'objet alternatif qui en est issu puisse s'y synchroniser.

Gestion de versions temporelles d'objets

5 En variante on peut mettre en œuvre pour les objets un mécanisme (classique) de gestion temporelle des versions, les contributions successives sur un même objet étant alors vues comme ses versions temporelles. Dans le cas où un contributeur d'une version temporelle précédente d'un objet refuse la version temporelle courante (la plus élaborée) d'un objet, il la fait passer dans une version²⁰³ étiquetée d'une paire composée du même ERC et de l'utilisateur qui a contribué la version refusée.

10 Les objets de remplacement peuvent ainsi circuler le long des transclusions comme déjà décrit notamment à la section « Propagation des contributions dans les branches ».

Dans le cas où l'outil de manipulation d'informations est muni d'une fonctionnalité de gestion de versions, prenons par exemple le cas de Wikipedia, on peut mettre en œuvre le procédé suivant :

15 Lorsque le contributeur d'une version précédente d'une page de Wikipedia visite la version courante d'une page, le système lui présente, sous forme d'objets de remplacement, les modifications qui ont eu lieu sur cette page par rapport à la dernière version qu'il avait lui-même contribué.

20 L'utilisateur peut alors refuser certains objets de remplacement, et ces derniers sont alors mis dans d'autres branches, c'est-à-dire qu'ils forment d'autres versions d'objets multi-versions.

Avantageusement, un tel procédé peut être mis en œuvre par une simple extension du navigateur Web (et bien entendu on doit prévoir aussi un serveur de collaboration), ceci indépendamment du serveur de Wikipedia. Dans ce cas, seules les contributions des membres d'ERC sont présentées comme objets de remplacement, et deviennent des objets alternatifs
25 lorsqu'ils sont refusés. Le serveur de Wikipedia lui-même (qui fournit les versions temporelles des pages de Wikipedia) peut jouer le rôle de serveur de contenu. Les utilisateurs qui ne sont pas membres d'ERC (ou qui n'ont pas l'extension du navigateur Web) continuent à utiliser Wikipedia comme avant.

Architecture globale de mise en oeuvre

30 Les figures 142 et 143 présentent une architecture comprenant le serveur de clés décrit ci-dessus ainsi qu'un serveur de transformations qui met en œuvre les transformations associées aux transclusions comme décrit précédemment.

Tout d'abord la figure 142 illustre le procédé de mise à jour de contenus par des utilisateurs.

35 Dans l'exemple présenté dans cette figure, un contenu a, qui se trouve dans un premier serveur de contenu, est transclus par une transclusion T1 pour donner le résultat est T1:a qui est stocké dans un deuxième serveur de contenu, puis ce dernier est lui-même transclus par la transclusion T2 pour donner T2:T1:a dans un troisième serveur de contenu. A est par ailleurs transclus par la transclusion T3 pour engendrer le résultat T3:a.

40 A noter que les résultats de transclusions qui sont ici présentés comme étant stockés dans des serveurs de contenus différents, peuvent aussi bien se trouver dans un même serveur de contenu²⁰⁴ et peuvent être des versions d'un même objet multi-versions, cela ne changerait en rien les principes de fonctionnements présentés ici. On décrit donc ici une architecture de mise en œuvre incluant le cas des objets multi-versions.

²⁰³ (au sens des versions d'un objet multi-versions)

²⁰⁴ (ou d'ailleurs même sur le poste client de l'utilisateur)

A noter aussi que tous ces contenus se
comme décrit à la section précédente.

les serveurs de contenu sont chiffrés

Les flèches marquées (1) indiquent les accès des utilisateurs pour modifier respectivement a, T3:a et T1:a.

- 5 Le procédé comprend une étape, représentée par les flèches (1'), qui consiste à communiquer au serveur de transformation les modifications, ainsi que les clés générées suite à ces modifications (et qui ont permis de chiffrer les contenus modifiés en question).

Ensuite, en (2), le serveur de transformation communique

- au serveur de collaboration les identifiants des contenus modifiés,
- 10 • et au serveur de clé les associations clé – identifiant de contenu modifié.

- 15 Pour éviter que le serveur de collaboration ne propage des transclusions qui en réalité n'auront pas lieu²⁰⁵, en (3), le serveur de collaboration communique au serveur de transformation les identifiants des nouveaux résultats potentiels de transclusions²⁰⁶ et ce dernier lui retourne un ensemble filtré de résultats. En effet, le serveur de transformation est apte à appliquer les transformations associées aux transclusions (qui peuvent notamment être des filtres appliqués aux sources) lorsqu'ils ne font pas intervenir d'autres objets que les contenus modifiés qui lui ont été communiqués, et déterminer s'ils sont filtrés par ces transformations.

La figure 143 présente le procédé de consultation de contenus.

En (1), l'utilisateur²⁰⁷ obtient le contenu T2:T1:a du serveur de transformation.

- 20 En (2), ce dernier obtient du serveur de collaboration les adresses des contenus qui fournissent des mises à jour à effectuer sur T2:T1:a²⁰⁸, ici il s'agit donc des contenus a, T1:a et T3:a qui ont été modifiés comme illustré dans la figure 142.

En (3) le serveur de transformation obtient les clés desdits contenus à partir du serveur de clés.

- 25 En (4), le serveur de transformation applique les transformations sur les sources modifiées qu'il obtient des serveurs de contenus respectifs.

En (5), l'utilisateur enregistre le contenu T2:T1:a mis à jour et transformé, dans le serveur de contenu.

- 30 On peut mettre en œuvre plusieurs méthodes pour enregistrer des contenus modifiés. Tout d'abord les modifications peuvent être enregistrées dans une structure ne contenant que les contenus modifiés et des références aux sources. Ensuite, et ceci devient avantageux quand la quantité de modifications dépasse un certain seuil, on peut enregistrer toute la version courante du contenu en question.

- 35 En (4), dans le cas où seules les modifications de T2:T1:a sont enregistrées, le serveur de transformations obtient tout le contenu de la source « a » à partir du premier serveur de contenu (ainsi que les modifications des autres sources). En revanche, dans le cas où la version courante de T2:T1:a avait été enregistrée entièrement, au lieu de tout le contenu de a, seules les modifications non encore prises en compte de a sont obtenues du premier serveur de contenu.

²⁰⁵ Ceci peut être vu comme une optimisation car ces transclusions seraient de toutes façons filtrés lors de la fourniture de résultats dans le procédé de consultation de contenus, comme décrit plus bas.

²⁰⁶ En utilisant l'« algorithme en push »

²⁰⁷ (c'est-à-dire l'extension de l'outil de manipulation d'information)

²⁰⁸ Ces mises-à-jours incluent les modifications à l'amont et les contributions (objets de remplacement et alternatifs) qui peuvent aussi provenir de l'aval.

De plus, dans le cas où la version courante a avait été enregistrée entièrement, en variante du point (1), l'utilisateur peut obtenir du troisième serveur de contenu la dernière version courante du contenu T2:T1:a et les clés correspondantes au serveur de clés, et indépendamment obtenir ensuite du serveur de transformation les mises à jours à y appliquer.

5 Mesure des contributions

Calcul du crédit dont bénéficie un utilisateur

Il s'agit de mesurer combien un utilisateur donné contribue à un ERC, ainsi que dans quelle mesure ses contributions sont appréciées ou au contraire refusées par les autres utilisateurs.

- 10 On attribue donc à chaque utilisateur un crédit, initialement neutre. Différentes actions des utilisateurs vont altérer ces crédits.

Crédit propre

Le crédit propre d'un utilisateur est augmenté lorsqu'il apporte une contribution.

- 15 Lorsqu'un utilisateur change de branche (soit à cause d'un refus d'objet de remplacement ou par changement explicite), le dernier contributeur de chaque partie constituant l'objet précédemment affiché voit son crédit propre diminuer, et le dernier contributeur de chaque partie constituant l'objet sélectionné voit le sien augmenter. De plus un changement de branche dû à un refus doit provoquer une variation de crédit propre plus grande qu'un changement de branche manuel (car cette dernière opération ne signifie pas nécessairement que le contenu désélectionné soit de mauvaise qualité).

- 20 Afin d'obtenir des résultats plus précis on pondère ces changements de crédit propre par l'ampleur du contenu concerné (une petite contribution influence donc moins le crédit propre de son auteur).

De plus, on peut donner un poids plus grand aux actions des utilisateurs ayant un crédit (total) plus élevé.

- 25 Il faut cependant éviter qu'il existe des méthodes pour un utilisateur d'augmenter arbitrairement son crédit propre (seul ou grâce à des complices). Notamment, si le simple fait de contribuer un contenu permettait à un utilisateur d'augmenter son crédit propre, des utilisateurs pourraient simplement produire une grande quantité de données vides de sens et ainsi augmenter leur crédit. Voici des approches de solution :

- 30 Si le contenu contribué est refusé par de nombreux utilisateurs, la pénalité est plus grande que le crédit propre obtenu par sa création.

- 35 Le crédit gagné au moment de la création d'un contenu dépend d'une estimation du nombre d'utilisateurs qui le verront (afin d'éviter qu'un utilisateur obtienne du crédit en produisant du contenu que personne ne va lire et donc refuser). Un moyen d'estimer cette valeur est le nombre d'utilisateurs chez qui la branche contenant cet objet est sélectionnée. On peut également tenir compte des transclusions ayant cet objet ou son parent comme source afin de compter les utilisateurs qui verront une image de ce nouveau contenu dans un résultat de transclusion.

- 40 Enfin, (toujours en fonction du nombre d'utilisateur pouvant accéder à ce contenu) on fixe une limite maximale au crédit pouvant être obtenu par ajout de contenu. Au-delà de cette limite, contribuer de nouveaux contenus n'a pas d'effet sur le crédit de l'auteur.

Crédit d'approbation

Les utilisateurs contribuant des contenus dans un domaine suivi par peu d'autres utilisateurs ne doivent pas être trop pénalisés, car c'est le domaine qui est impopulaire et pas spécifiquement les contributions de ces utilisateurs. La méthode décrite ci-dessous montre comment des

5 utilisateurs peuvent augmenter le crédit des utilisateurs qu'ils approuvent.

On tient à jour pour toute paire d'utilisateur le « facteur d'approbation » du premier pour le second. Pour éviter que ceci ait un effet exagéré on limite également la quantité de crédit qu'un utilisateur peut créer pour un autre, en bornant ce facteur entre 0 et $1/n$, où n donne un ordre de grandeur du nombre d'utilisateurs allant voir une ressource.

10 Lorsqu'un utilisateur montre son intérêt pour une contribution (la façon la plus claire étant en en faisant une transclusion, mais simplement sélectionner la branche correspondante est également une indication d'intérêt, de même que simplement la visite de la ressource contenant l'objet concerné), le facteur d'approbation du visiteur pour l'auteur de la contribution augmente.

15 Afin de réguler les facteurs d'approbation, on peut par exemple les diminuer régulièrement, ce qui fait que lorsqu'un utilisateur cesse de montrer son intérêt pour les contributions d'un autre, son facteur d'approbation pour ce dernier va graduellement diminuer.

Soit donc x le crédit propre d'un utilisateur, et a_1, a_2, a_3 , etc les facteurs d'approbation de la part d'utilisateurs ayant x_1, x_2, x_3 , etc comme crédit. On suppose $x_1 < x_2 < x_3 < \text{etc}$. Pour

20 calculer le crédit effectif de l'utilisateur on procède comme suit.

Le crédit initial est $K=x$.

On effectue l'opération suivante successivement pour $i=1, i=2$, etc.

Si $K < x$ alors on remplace K par $K + (x_i - K) \cdot a_i$.

25 Ainsi, si le crédit du visiteur est supérieur au crédit de l'auteur de la contribution, le crédit du second est augmenté (en évitant bien sûr d'effectuer cette augmentation plus d'une fois pour un visiteur et une contribution).

Application du procédé de conduits de propagation de scores de pertinence

30 On peut adapter le procédé des conduits de propagation de scores de pertinence comme décrit dans WO05045698A2 à une structure de transclusions comme suit.

L'idée de base est que le score d'un objet est lié à ce qui est présent (visible, audible,...) dans cet objet (et si possible au sens de ce contenu). On va donc placer des conduits (tous de même grosseur) le long des transclusions, propageant le score depuis un objet de la source en direction de son image dans le résultat de la transclusion. En présence de transclusions

35 internes, les conduits sont placés suivant les flux d'information, c'est-à-dire que pour un résultat de transclusion n'est placé un conduit que depuis la source candidate la plus proche.

Lorsque la source et le résultat d'une transclusion ne sont pas dans la même branche, la grosseur du conduit est amoindrie, par exemple est fixée à la moitié de la grosseur d'un conduit ne changeant pas de branche, afin de refléter le fait que les objets de remplacement ne

40 changent pas de branche.

Afin de traiter convenablement les changements effectués dans un résultat de transclusion, on place également des conduits propageant le score de tous les objets en direction de leurs parents. Ceci aura pour effet que les parents auront un score tenant compte du score moyen de leurs enfants. Ainsi, si dans un résultat de transclusion des objets sont ajoutés, modifiés ou

45 supprimés, le score du résultat de transclusion sera affecté en conséquence.

Trois applications en sont les suivantes :

Lorsqu'un objet reçoit beaucoup de contributions alternatives, le système peut ainsi les trier par pertinence (et ne garder que les premiers), en calculant la proximité de chaque contribution avec le contenant²⁰⁹, ainsi qu'éventuellement en se basant sur le crédit du contributeur.

- 5 Deuxièmement, on peut placer sélectivement des contenus (tels que des publicités) dans un objet, en fonction de leur pertinence vis-à-vis de l'objet contenant (éventuellement en se basant également sur le crédit du contributeur).

- 10 Troisièmement, on peut aussi utiliser le procédé décrit dans WO-03 057648 pour sélectionner des annotations mises sur des ressources proches. Il est à noter que dans cette dernière application, on peut bénéficier de la structure des « positions » des parties homogènes maximales (décrite plus haut) pour y placer des informations permettant de déterminer comment présenter dans une ressource des annotations qui figurent dans une ressource proche.

15	Définitions préliminaires.....	10
	Objet, Ressource	10
	ERC, Serveur collaboratif.....	11
	Utilisateur	11
	Description	12
20	Introduction.....	12
	Illustration du principe du procédé	12
	Versions d'une ressource collaborative, branches.....	13
	Visionner les versions alternatives d'une ressource collaborative	15
	Dérivation	16
25	Catégorisation des versions de ressources collaboratives	17
	Modifications structurelles, créneaux	17
	Gestion des incohérences.....	19
	Fusion de branches	20
	Cas de la ressource collaborative unique.....	20
30	Transclusions	21
	Mise en oeuvre.....	21
	Architecture	21
	Structures de données.....	22
	Contraintes	23
35	Algorithmes	23
	Construction de la structure à présenter pour une version de la ressource collaborative	23
	Changement de branche active	24
	Réaction à une modification	24
40	Traitement d'une création d'objet	24
	Traitement d'un refus	25
	Gestion des opérations concurrentes	26
	Restrictions de propagation	26
	Protection en lecture.....	26
45	Protection en écriture.....	26
	Propagations unidirectionnelles	27
	Variante : Versions de ressources comme Transclusions	27
	Illustration du principe du procédé	27

²⁰⁹ (éventuellement en prenant en compte le contexte de navigation de l'utilisateur).

	105	
	Procédé de création d'objets les plus é	28
	Création de ressource dérivée	28
	Les modifications sont proposées à l'amont	29
	Exemple	30
5	Situation initiale de l'exemple	30
	Procédé de détermination de l'objet de remplacement et des objets alternatifs	31
	Ignorance ou acceptation (fig. 26)	32
	Refus (fig. 27 à 32)	32
	Dérivations multiples d'une même ressource (fig. 33 à 44)	33
10	Variante à la Règle 3 (fig. 45)	34
	Remplacement/Enrichissement de Grains (fig. 46 à 49)	34
	Créneaux (fig. 50)	37
	Procédé de tri d'objets alternatifs par notation (rating)	38
	Procédé de tri d'objets alternatifs par pertinence contextuelle	39
15	Procédé de remplacement par co-identification	39
	Procédé de transclusion	40
	Identifiant	41
	Transclusion	41
	Transclusion implicite, transclusion interne	44
20	Equivalence d'identifiants	44
	Calcul des sources d'un objet	44
	Origine d'une information	44
	Distance de sources	45
	Première méthode	45
25	Calcul de la proximité d'un objet par rapport à un autre	45
	Deuxième méthode	46
	Procédé de transclusion en « pull »	46
	Calcul d'un objet – algorithme local	46
	Calcul d'une structure d'objets – algorithme global	47
30	Procédé de transclusion en « push »	47
	Structures mixtes	49
	Architecture centralisée ou en cascade	51
	Propagation des contributions dans les branches	52
	Variante	52
35	Comportement des grains en présence de transclusions	53
	Compteurs d'activité	54
	Objets conteneurs, Transclusion-fusion, Placement automatique	55
	Conditions associées aux transclusions	57
	Architecture de personnalisation	58
40	Remplacements/Enrichissements comme propositions d'acceptation	61
	Modes d'utilisation des ERC	62
	Recommandation de Ressources et d'ERC	63
	Avantages	63
	Avantages pour les procédés de publication	64
45	Rédaction collective d'une publication future	64
	Avantages pour diffuser des contenus publicitaires	65
	Avantages pour les « wikis »	65
	Le Web en lecture-écriture	66
	Liens inverses	66
50	Méta-données de contrainte de diffusion	66
	Avantage de contourner la censure	67
	Extensions aux protocoles de syndication actuels	68

	Interopérabilité.....	68
	PSE.....	69
	Spécification XML.....	70
	Élément <pse:part> dans <item> ou <channel>.....	70
5	Scenarios PSE.....	70
	Item transclus dans un canal.....	70
	Représentation XML abstraite.....	70
	Item transclus au sein d'un autre item.....	72
	Représentation XML abstraite.....	72
10	Partie d'item transcluse au sein d'un autre item.....	74
	Représentation XML abstraite.....	74
	Transclusions internes et implicites.....	74
	ASE.....	76
	Spécification XML.....	76
15	Exemple.....	77
	Transclusions et Contributions vers l'amont (PSE avec ASE).....	78
	Exemple.....	78
	Représentation XML abstraite.....	78
	Serveur Collaboratif pour PSE/ASE.....	81
20	Transclusion avec transformations.....	83
	Exemples de transformations.....	85
	Exemples introductifs : remplacement et anti-doublons.....	85
	Avec modifications locales.....	85
	Filtres de sélection par rapport à un critère.....	85
25	Remplacement de transformations.....	86
	Principes d'intégration à un tableur tel que Microsoft Excel.....	87
	Objets multi-versions.....	90
	Exemple d'objet multi-versions.....	91
	Interface utilisateur.....	91
30	Mise en œuvre à l'aide de transclusions.....	93
	Objets multi-versions imbriqués.....	93
	Approches de mise en œuvre du contrôle d'accès.....	94
	Clés pour ERC en lecture et en écriture.....	94
	Serveur de chiffrement/déchiffrement.....	94
35	Chiffrement des objets multi-versions.....	96
	Possibilités de transfert.....	98
	Exemple.....	98
	Objets de remplacement dans les objets multi-versions.....	99
	Gestion de versions temporelles d'objets.....	100
40	Architecture globale de mise en oeuvre.....	100
	Mesure des contributions.....	102
	Calcul du crédit dont bénéficie un utilisateur.....	102
	Crédit propre.....	102
	Crédit d'approbation.....	103
45	Application du procédé de conduits de propagation de scores de pertinence.....	103

107
REVNS

1. Procédé pour accéder à une ressource dans un environnement informatique, la ressource comprenant un ensemble d'objets, caractérisé en ce que l'environnement informatique est apte à mémoriser en association avec au moins un objet de ladite ressource au moins un objet modifié, en ce que ledit environnement informatique comprend des moyens aptes à mémoriser en association avec un tel objet une information de degré d'élaboration, et en ce que le procédé comprend les étapes suivantes :
- identifier, pour chaque objet de la ressource auquel correspond au moins un objet modifié, à partir des informations de degré d'élaboration, au moins une version la plus élaborée dudit objet,
 - assembler les objets ainsi identifiés en vue de leur affichage dans la ressource.
2. Procédé selon la revendication 1, caractérisé en ce que les informations de degré d'élaboration d'un objet découlent de la chronologie des modifications de l'objet, en ce qu'il est prévu lors de la modification d'un objet par un utilisateur une étape permettant à d'autres utilisateurs de refuser l'objet modifié, et en ce que l'étape d'assemblage est réalisée avec une version de l'objet choisie en fonction de l'existence ou non de refus.
3. Procédé selon la revendication 2, caractérisé en ce qu'il comprend, lorsqu'une version modifiée d'un objet est refusée, la création de deux branches contenant deux versions alternatives de l'objet, à savoir la version modifiée et la version non modifiée.
4. Procédé selon la revendication 3, caractérisé en ce que l'étape d'assemblage d'une ressource incluant une certaine version de l'objet en question est effectuée en signalant l'existence d'une version ou de versions alternatives dudit objet.
5. Procédé selon la revendication 4, caractérisé en ce qu'il comprend en outre une étape de sélection, par un utilisateur accédant à une version de la ressource, d'une certaine version alternative de l'objet.
6. Procédé selon la revendication 5, caractérisé en ce que des versions plus élaborées des objets sont formées indépendamment pour les différents objets alternatifs dans chacune des branches.
7. Procédé selon l'une des revendications 3 à 6, caractérisé en ce qu'il comprend en outre la mise en œuvre de moyens de calcul de scores de pertinences des différentes versions alternatives des objets pour présenter sélectivement les versions les plus pertinentes.
8. Procédé selon la revendication 2, caractérisé en ce que plusieurs versions d'une ressource peuvent être engendrées par des utilisateurs, et en ce que dans l'étape d'assemblage, on propose par défaut la version de l'objet en fonction de la version de la ressource.
9. Procédé selon la revendication 1, caractérisé en ce qu'il est prévu une étape d'inhibition de l'utilisation d'un objet dans l'assemblage d'une ressource si une condition donnée relative à cet objet ou à un autre objet de la ressource à assembler n'est pas remplie.
10. Procédé selon la revendication 9, caractérisé en ce que le contenu d'un objet dont l'utilisation peut être inhibée est crypté, et en ce que l'étape d'inhibition comprend l'inaccessibilité à une clé de décryptage.

11. Procédé selon la revendication 9¹⁰⁸ caractérisé en ce que la condition donnée est une condition de présence d'au moins un objet d'une autre ressource.
- 5 12. Procédé selon la revendication 9 ou 10, caractérisé en ce que la condition donnée est une condition de non-modification de l'objet présent.
13. Procédé selon la revendication 1, caractérisé en ce qu'il est mis en œuvre dans un environnement informatique comprenant :
- 10 - au moins un serveur de contenus apte à fournir des contenus d'objets pouvant être assemblés pour former des ressources,
- un gestionnaire de collaboration apte à gérer les versions des ressources et les versions des objets qu'elles contiennent,
- 15 - au niveau d'au moins un poste utilisateur, une extension (plug-in) d'un outil de manipulation d'objets et de ressources, apte à communiquer avec le gestionnaire de collaboration,
- et en ce qu'il comprend une étape de vérification de l'authenticité de l'extension par le gestionnaire de collaboration.
- 20 14. Procédé selon la revendication 1, caractérisé en ce que l'information de degré d'élaboration est apte à prendre l'une parmi deux valeurs, à savoir « le plus élaboré » ou non.
15. Procédé selon la revendication 14, caractérisé en ce que l'étape d'identification comprend la recherche d'objets (OR ; OA) dont le degré d'élaboration possède la valeur « le plus élaboré » parmi les objets aval successivement modifiés à partir de l'objet en question de la ressource accédée.
- 25 16. Procédé selon la revendication 1, caractérisé en ce que l'étape d'identification comprend la recherche d'objets de degré d'élaboration supérieur à une limite parmi les objets aval successivement modifiés à partir de l'objet en question de la ressource accédée.
- 30 17. Procédé selon l'une des revendications 1 à 16, caractérisé en ce que la recherche inclut la recherche d'objets de degré(s) d'élaboration le(s) plus élevé(s) en parcourant les objets amont par modification desquels l'objet en question a été obtenu.
- 35 18. Procédé selon l'une des revendications 1 à 17, caractérisé en ce que, dans le cas où l'étape d'identification a permis d'identifier plusieurs objets les plus élaborés, il est prévu dans l'étape d'assemblage des objets une sous-étape d'ajout à la ressource d'une information signalant l'existence de tels plusieurs objets (OA).
- 40 19. Procédé selon la revendication 18, caractérisé en ce que l'objet sélectionné pour l'assemblage dans la ressource est l'objet aval le plus élaboré le plus proche dans la succession.
- 45 20. Procédé selon la revendication 18 ou 19, caractérisé en ce qu'en réponse à des instructions de changement d'objet reçues de l'utilisateur, l'étape d'assemblage est apte à utiliser successivement pour l'assemblage des objets les plus élaborés triés selon un autre critère tel qu'une notation.
- 50 21. Procédé selon l'une des revendications 18 à 20, caractérisé en ce qu'il est apte, en réponse à une instruction de sélection par un utilisateur, à utiliser pour l'assemblage un objet désigné par ladite instruction.

22. Procédé selon les revendications 1 à 21, en combinaison, caractérisé en ce qu'il comprend en outre une étape d'ajustement de la notation de l'objet en fonction d'une instruction de sélection de cet objet par l'utilisateur.

5 23. Procédé selon la revendication 18 ou 19, caractérisé en ce que l'étape d'assemblage est apte à utiliser pour l'assemblage un objet sélectionné selon un autre critère tel qu'une notation parmi tous les objets les plus élaborés.

10 24. Procédé selon l'une des revendications 1 à 23, caractérisé en ce qu'il est mis en œuvre en combinaison avec un serveur de contenus préexistants doté de moyens de gestion chronologique de versions des objets et en ce que l'étape d'identification de la version la plus élaborée d'un objet est mise en œuvre à partir d'informations de versions fournis par lesdits moyens de gestion chronologique et d'informations de contribution d'un utilisateur accédant à la ressource aux différentes versions, et en ce qu'il comprend en outre une étape de
15 présentation de la version courante de l'objet dans le serveur de contenus.

20 25. Procédé mis en œuvre en association avec des moyens de mémorisation accessibles dans un environnement informatique pour maintenir à jour des graphes d'objets susceptibles d'être assemblés pour former des ressources accessibles dans l'environnement, caractérisé en ce que les moyens de mémorisation sont aptes à mémoriser en association avec ces objets une information de degré d'élaboration, et en ce que le procédé comprend les étapes suivantes :

- détecter la modification d'un objet par un utilisateur,
- identifier des propriétaires de ressources contenant des objets correspondants à l'amont,
25 - diriger vers ces propriétaires une proposition d'acceptation ou de refus de cette modification,
- attendre une réponse des propriétaires,
- en fonction du contenu d'au moins la première réponse, ajuster les informations de degré d'élaboration des objets.

30 26. Procédé selon la revendication 25, caractérisé en ce que l'information de degré d'élaboration est apte à prendre l'une parmi deux valeurs, à savoir « le plus élaboré » ou non.

35 27. Procédé selon l'une des revendications 25 et 26, caractérisé en ce qu'il comprend en outre une étape consistant, en réponse à la détection de la modification d'un objet, à ajuster temporairement les informations de degré d'élaboration des objets, cet ajustement pouvant être contrarié ou confirmé à l'issue d'au moins la première réponse des propriétaires.

40 28. Procédé selon l'une des revendications 25 à 27, caractérisé en ce que, lors d'une première réponse d'acceptation d'un propriétaire, on applique la modification dans la ressource dudit propriétaire.

45 29. Procédé selon l'une des revendications 25 à 28, caractérisé en ce que, lors d'une réponse de refus d'un propriétaire, alors l'étape d'ajustement donne à l'objet immédiatement en amont de l'objet modifié un degré d'élaboration le plus élevé.

50 30. Procédé selon la revendication 29, caractérisé en ce qu'il comprend en outre, lors d'une réponse de refus, l'ajout au graphe d'une branche contenant une version alternative de l'objet constituée par l'objet incluant la modification.

31. Procédé selon la revendication 29 ou 30, caractérisé en ce qu'il comprend en outre, lors d'une réponse de refus d'une modification effectuée sur un objet qui n'est pas le plus élaboré

et en cas d'incompatibilité de la modification 110 1 moins objet plus élaboré, l'ajout au graphe d'une branche contenant une version alternative de l'objet constituée par l'objet incluant la modification ou contenant le ou les objets incompatibles et les objets modifiés à partir d'eux.

5

32. Procédé de création d'une nouvelle ressource à partir d'une ressource existante à laquelle on a accédé dans un environnement informatique, caractérisé en ce qu'il comprend les étapes suivantes :

- 10 - détecter la modification d'un objet de la ressource existante par un utilisateur,
- en réponse à cette détection, créer une nouvelle ressource présentant le même contenu que la ressource existante,
- appliquer les modifications à l'objet de ladite autre ressource qui correspond à l'objet de la ressource existante pour lequel la modification a été détectée.

15 33. Procédé selon la revendication 32, caractérisé en ce que l'étape de création d'une nouvelle ressource est effectuée par transclusion à partir d'informations associées à la ressource existante.

20 34. Procédé mis en œuvre en association avec des moyens de mémorisation accessibles dans un environnement informatique pour maintenir à jour des graphes d'objets susceptibles d'être assemblés pour former des ressources accessibles dans l'environnement, caractérisé en ce que les moyens de mémorisation sont aptes à mémoriser en association avec chacun de ces objets une information de degré d'élaboration, et en ce que le procédé comprend les étapes suivantes :

- 25 - détecter la modification d'un objet par un utilisateur,
- en réponse à cette détection, créer une nouvelle ressource présentant le même contenu que la ressource existante,
- appliquer les modifications à l'objet de ladite autre ressource qui correspond à l'objet de la ressource existante pour lequel la modification a été détectée,
- 30 - identifier des propriétaires de ressources contenant des objets correspondants à l'amont,
- diriger vers ces propriétaires une proposition d'acceptation ou de refus de cette modification,
- attendre une réponse des propriétaires,
- 35 - en fonction du contenu d'au moins la première réponse, ajuster les informations de degré d'élaboration des objets.

40 35. Procédé pour propager des informations standard de description d'objet entre différents objets susceptibles d'être assemblés pour former des ressources visualisables dans une interface utilisateur au sein d'un environnement informatique, caractérisé en ce qu'à chaque objet sont associées des méta-données de description, et en ce que le procédé comprend les étapes suivantes :

- 45 - l'entrée dans un mode de propagation à l'initiative de l'utilisateur,
- la sélection d'un objet dont on souhaite emprunter les méta-données de description,
- la sélection d'au moins un autre objet pour lui affecter les mêmes méta-données de description.

36. Procédé selon la revendication 35, caractérisé en ce que l'interface utilisateur est adaptée pour masquer à l'utilisateur les méta-données de description.

50

37. Procédé pour rendre accessible à sein d'un environnement informatique une ressource contenant au moins un objet dérivé d'une autre ressource, caractérisé en ce qu'il comprend les étapes suivantes :

- détecter une requête d'un utilisateur en vue de modifier l'objet,
- 5 - en réponse à cette détection, engendrer dans l'environnement informatique une requête auprès d'un propriétaire de ladite autre ressource en vue d'autoriser l'accessibilité de l'objet modifié,
- en fonction de la réponse reçue du propriétaire, sélectivement autoriser ou non l'accessibilité.

10

38. Procédé selon la revendication 37, caractérisé en ce qu'une requête commune est engendrée pour une pluralité d'objets dérivés de ressources appartenant au même propriétaire.

15

39. Ressource informatique apte à permettre la mise en œuvre du procédé selon la revendication 37 ou 38, comprenant un ensemble d'objets et accessible au travers d'un environnement informatique, caractérisée en ce qu'en association avec au moins certains des objets est mémorisée une méta-donnée de contrainte de diffusion du contenu de l'objet, permettant de sélectivement contrôler sa modification et/ou son accessibilité par des tiers au sein de ressources dérivées.

20

40. Ressource selon la revendication 39, caractérisée en ce que les méta-données de contraintes de diffusion comprennent au moins une méta-donnée choisie dans le groupe comprenant les méta-données d'autorisation/interdiction de modification, des méta-données d'autorisation/interdiction d'accès aux tiers des objets non modifiés ou modifiés, et des méta-données de contact d'une autorité décisionnaire.

25

41. Ressource selon la revendication 39 ou 40, caractérisé en ce qu'elle est structurée sous forme d'une arborescence d'objets, en ce qu'en association avec au moins certains nœuds de l'arborescence, une méta-donnée de contrainte de diffusion est mémorisée, et en ce qu'une méta-donnée de contrainte de diffusion au niveau d'un nœud s'applique aux nœuds enfants du nœud en question.

30

42. Procédé pour rendre accessible à des tiers au sein d'un environnement informatique une ressource contenant au moins un objet dérivé d'une autre ressource, caractérisé en ce qu'aux objets sont susceptibles d'être associées, de façon mémorisée dans l'environnement informatique, des méta-données de contrainte de diffusion et ces méta-données étant susceptibles de contenir une identification d'un propriétaire d'un objet à partir duquel l'objet précité doit être dérivé, et en ce que le procédé comprend les étapes suivantes :

35

- détection du fait qu'un objet contenu dans une ressource à laquelle un utilisateur souhaite accéder est dérivée d'une autre ressource,
- 40 - détection de l'existence dans l'environnement informatique de méta-données de contrainte de diffusion de l'objet en question,
- dans l'affirmative, lecture desdites méta-données,
- détection de l'existence dans lesdites méta-données d'une identification d'une
- 45 autorité décisionnaire,
- dans l'affirmative, envoi à ladite autorité ainsi identifiée d'une requête d'autorisation d'accessibilité à des tiers, et
- dans le cas où une acceptation de ladite requête est reçue, mise à disposition de la ressource auxdits tiers.

50

43. Procédé selon la revendication 42, caractérisé en ce que la mise à disposition de la ressource est une publication.

44. Procédé selon la revendication 42 ou 43, caractérisé en ce qu'il comprend l'étape intermédiaire consistant, entre l'envoi de la requête et la réception d'une autorisation ou d'un refus, une étape d'assemblage provisoire de la ressource avec un objet formé d'un extrait de l'objet que l'on souhaite dériver.

45. Procédé selon la revendication 44, caractérisé en ce que, lorsque l'objet est un objet texte, ledit extrait est élaboré en fonction de critères juridiques de droit de citation.

46. Procédé pour estimer quantitativement l'activité d'objets dans un environnement informatique, cet environnement permettant l'accès à des ressources réalisées par assemblage d'objets, au moins certains des objets pouvant être engendrés par dérivation d'objets préexistants et modification des objets ainsi dérivés, procédé caractérisé en ce qu'il comprend les étapes suivantes :

- identifier des objets les plus amont destinés à être dérivés, de telle sorte que cette identification se propage aux objets dérivés de ceux-ci,
- compter les nombres de dérivations des objets les plus amont et des objets qui en ont été dérivés.

47. Procédé selon la revendication 46, caractérisé en ce qu'il comprend en outre une étape de comptage groupé des consultations des objets les plus amont et des objets qui en ont été dérivés.

48. Procédé selon la revendication 46 ou 47, caractérisé en ce que les objets les plus amont sont des objets à caractère publicitaire.

49. Procédé selon l'une des revendications 46 à 48, caractérisé en ce que les objets les plus amont sont aptes à être dérivés sans pouvoir être modifiés.

50. Environnement informatique pour mettre en œuvre un procédé selon l'une des revendications 1 à 38 ou 42 à 49, caractérisé en ce qu'il comprend une pluralité de serveurs reliés en maillage, aptes à contenir des informations de dérivation entre objets et de marquage de degrés d'élaboration, chaque serveur étant apte à recevoir une requête en vue d'indiquer des identifiants d'objets sur la base desdites informations, et étant apte à rediriger une telle requête vers un autre serveur dans le cas où il n'est pas apte à répondre à ladite requête.

51. Procédé pour gérer des ressources à contenus variables dans un environnement informatique, chaque ressource comprenant un ensemble d'objets et au moins certains de ces objets étant présentés dans une ressource en formant un objet transclus individuellement ou en tant que partie d'un objet transclus plus grand à partir d'un objet source, via le cas échéant un ou plusieurs objets transclus intermédiaires, de telle sorte qu'une modification apportée à l'objet source puisse se propager jusqu'audit objet transclus, le procédé étant caractérisé :

- en ce que l'environnement informatique est apte à mémoriser des informations de transclusion basées sur des identifiants d'objets, informations à partir desquelles des objets sources ont la possibilité d'être présentés dans des objets transclus, et
- en ce que le procédé comprend l'étape consistant, lors de la modification d'un objet, à engendrer ou à mettre à jour, pour chaque objet transclus à partir dudit objet, un identifiant d'objet source à prendre en compte pour cet objet transclus, en fonction d'une information de distance entre l'objet modifié et ledit objet transclus.

52. Procédé pour gérer des ressources à contenus variables dans un environnement informatique, chaque ressource comprenant un ensemble d'objets et au moins certains de ces

objets étant présentés dans une ressource objet transclus individuellement ou en tant que partie d'un objet transclus plus grand à partir d'un objet source, via le cas échéant un ou plusieurs objets transclus intermédiaires, de telle sorte qu'une modification apportée à l'objet source puisse se propager jusqu'audit objet transclus, le procédé étant caractérisé :

5 - en ce que l'environnement informatique est apte à mémoriser des informations de transclusion basées sur des identifiants d'objets, informations à partir desquelles des objets sources ont la possibilité d'être présentés dans des objets transclus, et

10 - en ce que le procédé comprend l'étape consistant, lors de la présentation d'un objet, à déterminer, pour chaque objet transclus à partir dudit objet, un identifiant d'objet source à prendre en compte pour cet objet transclus, en fonction d'une information de distance entre l'objet modifié et ledit objet transclus.

15 53. Procédé selon la revendication 51, caractérisé en ce que ladite étape, lors de la modification d'un objet, identifie cet objet comme sa propre source, avec une distance nulle.

54. Procédé selon l'une des revendications 51 à 53, caractérisé en ce que les ressources comprennent des structures arborescentes d'objets, et en ce qu'il comprend en outre une étape consistant, pour une transclusion d'un objet englobant des objets descendants dont l'un est transclus à partir de l'autre, à définir pour la transclusion entre objets descendants de l'objet englobant transclus, une valeur de distance entre ces objets plus courte.

25 55. Procédé selon l'une des revendications 51 à 54, caractérisé en ce que lesdits modifications peuvent concerner des attributs différents desdits objets, et en ce que l'étape de génération ou de mise à jour d'objet source est effectuée séparément pour chacun des attributs, un objet pouvant ainsi avoir des objets sources différents selon l'attribut considéré.

30 56. Procédé selon la revendication 55, caractérisé en ce que les attributs différents comprennent au moins deux attributs parmi des attributs de contenu, de méta-données, de position et de transformation.

35 57. Procédé pour gérer des ressources à contenus variables dans un environnement informatique, chaque ressource comprenant un ensemble d'objets et au moins certains de ces objets étant présentés dans une ressource en formant un objet transclus individuellement ou en tant que partie d'un objet transclus plus grand à partir d'un objet source, via le cas échéant un ou plusieurs objets transclus intermédiaires, de telle sorte qu'une modification apportée à l'objet source puisse se propager jusqu'audit objet transclus, caractérisé en ce qu'il comprend, selon la fréquence des modifications apportées aux objets et des présentations d'objets, la mise en œuvre soit du procédé selon la revendication 51, soit du procédé selon la revendication 52.

40 58. Procédé selon la revendication 57, caractérisé en ce que la sélection du procédé selon la revendication 46 est automatiquement sélectionnée quand l'environnement de l'objet source n'est pas apte à engendrer ledit identifiant d'objet source.

45 59. Procédé selon la revendication 57 ou 58, caractérisé en ce que la sélection du procédé mis en œuvre est effectuée transclusion par transclusion.

60. Procédé selon l'une des revendications 51 à 59, caractérisé en ce qu'il est mis en œuvre dans des moyens serveurs centralisés aptes à fournir les identifiants d'objets sources.

50 61. Procédé selon l'une des revendications 51 à 59, caractérisé en ce qu'il est mis en œuvre dans des moyens de traitement en cascade aptes à fournir les identifiants d'objets sources par parcours de proche en proche selon les transclusions.

62. Procédé selon l'une des revendications 51 à 61, caractérisé en ce qu'au moins à certaines transclusions sont associées des informations de transformation de l'objet, ces informations désignant chacune une fonction de transformation prédéterminée à appliquer à l'objet en tant qu'étape initiale de la transclusion.

63. Procédé selon la revendication 62, caractérisé en ce qu'une transformation est apte à être appliquée sur des attributs d'objet tels que des attributs de contenu, de position, de méta-données et de transformations portant sur des transclusions internes aux objets.

64. Procédé selon l'une des revendications 51 à 63, caractérisé en ce que les ressources sont des feuilles de calcul et en ce que les objets sont des cellules ou groupes de cellules de la feuille.

65. Procédé selon la revendication 64, caractérisé en ce que les formules associées aux cellules d'un groupe de cellules transclus sont considérées comme des transclusions internes au groupe.

66. Procédé selon la revendication 65, caractérisé en ce que l'obtention d'une transclusion interne dans un groupe de cellules transclus comprend l'application au programme de feuille de calcul d'instructions de copier/coller à partir d'une transclusion interne donnée du groupe de cellules source.

67. Procédé pour gérer un ensemble d'objets d'informations dans un environnement informatique, certains objets étant des objets conteneurs susceptibles de contenir d'autres objets, et l'environnement informatique comprenant une première structure d'objets et une deuxième structure d'objets transcluse à partir de la première, caractérisé en ce que :

- au moins certains objets de la première structure possèdent des propriétés ;
- au moins certains objets conteneurs de la deuxième structure possèdent des contraintes,

et en ce que le procédé comprend l'étape consistant, lors de l'opération de transclusion de la première structure vers la première structure, à présenter les objets de la première structure dans les objets conteneurs de la deuxième structure fonction de l'adéquation des propriétés des objets de la première structure avec les contraintes des objets conteneurs de la deuxième structure.

68. Procédé selon la revendication 67, caractérisé en ce qu'il comprend en outre l'étape consistant, lors de la modification d'une propriété d'un objet de la première structure, à vérifier l'adéquation de la propriété modifiée avec les contraintes des objets conteneurs de la deuxième structure, et à déplacer le cas échéant l'objet correspondant présenté dans la deuxième structure.

69. Procédé selon la revendication 67 ou 68, caractérisé en ce qu'il comprend en outre l'étape consistant, lors de la modification d'une contrainte d'un objet conteneur de la deuxième structure, à vérifier l'adéquation des propriétés des objets présentés dans cet objet conteneur avec la contrainte modifiée de celui-ci, et à déplacer le cas échéant le ou les objets considérés.

70. Procédé pour modifier des contenus constitués par des flux de données balisés auxquels on accède selon un protocole de syndication, chaque contenu étant accessible sous forme d'un canal qui comprend plusieurs éléments d'information possédant chacun un identifiant unique, caractérisé en ce qu'il comprend les étapes suivantes :

115

- délimitation d'un élément d'info
- application de la modification à l'élément d'information ou à la partie d'élément,
- élaboration d'un contenu contenant d'une part l'élément d'information ou partie d'élément modifié délimité par des balises spécifiques, et d'autre part l'élément d'information d'origine délimités par ses balises d'origine,
- de sorte à éviter une perte d'information lorsque le flux est traité selon un protocole qui ne gère pas lesdites balises spécifiques.
71. Procédé selon la revendication 70, caractérisé en ce qu'à l'élément d'information ou partie d'élément modifié est associée une information indiquant s'il remplace l'élément d'information ou partie d'élément d'origine ou s'il constitue une alternative à l'élément d'information ou partie d'élément d'origine, et en ce qu'il comprend dans ce dernier cas une étape de signalisation de l'existence de l'alternative.
72. Procédé selon l'une des revendications 70 et 71, caractérisé en ce qu'il comprend en outre une étape consistant, en réponse à la modification d'un élément d'information ou d'une partie d'élément, à signaler cette modification à un serveur d'où provient le flux de données.
73. Procédé selon l'une des revendications 1 à 38, 42 à 49 et 51 à 72, caractérisé en ce que les ressources sont des flux de données balisés auxquels on accède selon un protocole de syndication, et en ce que les objets desdites ressources sont des éléments d'information desdites ressources ou des parties desdits éléments d'information.
74. Procédé pour transformer une première ressource comprenant une structure arborescente d'objets en une deuxième ressource comprenant une structure arborescente d'objets, sur la base de transformations pouvant être effectuées sur des objets situés à des niveaux différents de l'arborescence, caractérisé en ce que, lorsqu'il existe une première transformation sur un certain objet englobant au moins un objet descendant et une seconde transformation sur ledit objet descendant, la seconde transformation est exécutée prioritairement à la première transformation.
75. Procédé pour l'accès par un utilisateur à des objets d'informations engendrés par un système collaboratif où différents groupes d'utilisateurs peuvent contribuer à former respectivement différentes versions d'objets à partir d'objets de départ communs, caractérisé en ce que l'utilisateur est susceptible d'appartenir simultanément à différents groupes, et en ce qu'il comprend les étapes consistant :
- en fonction d'un identifiant d'un des groupes sous lequel l'utilisateur réalise l'accès, présenter les versions correspondantes des objets partir des objets de départ,
 - en fonction d'une information de mode d'accès contributif/non contributif par l'utilisateur, prendre en compte ou non dans le système collaboratif au niveau du groupe considéré les actions modificatives subséquentes de l'utilisateur sur les versions d'objets auxquelles il a accédé.
76. Procédé selon la revendication 75, caractérisé en ce que, en mode d'accès non contributif lors de l'accès sous un groupe donné, les actions modificatives peuvent être prises en compte par défaut dans d'autres groupes.
77. Procédé pour l'accès par des utilisateurs à des objets d'informations engendrés par un système collaboratif où différents utilisateurs peuvent contribuer à former respectivement différentes versions d'objets, et où les utilisateurs peuvent sélectivement accepter ou refuser des versions, caractérisé en ce qu'il comprend des moyens de comptabilisation des

contributions des utilisateurs et des acc
d'objets contenant ces contributions.

par d'autres utilisateurs des versions

- 5 - en fonction d'une information de mode d'accès contributif/non contributif par l'utilisateur, prendre en compte ou non dans le système collaboratif au niveau du groupe considéré les actions modificatives subséquentes de l'utilisateur sur les versions d'objets auxquelles il a accédé.

10 78. Procédé pour rendre accessible à un groupe d'utilisateurs au sein d'un environnement informatique une ressource contenant une pluralité d'objets susceptibles d'être modifiés par les utilisateurs pour obtenir différentes versions desdits objets, caractérisé en ce qu'il comprend :

- 15 - pour au moins certains objets, affecter aux différentes versions desdits objets des clés de chiffrement/déchiffrement différentes,
- prévoir des versions desdits objets chiffrées avec les clés respectives,
15 - lors de l'accès à la ressource par un utilisateur, effectuer le déchiffrement des versions d'objets chiffrées appartenant à la ressource à l'aide des clés de déchiffrement respectives,
- lors la modification d'un tel objet par un utilisateur, créer une nouvelle version de l'objet, à laquelle une nouvelle clé est affectée.

20 79. Procédé selon la revendication 78, caractérisé en ce qu'il comprend une étape de fourniture sélective des clés de déchiffrement des objets aux utilisateurs en fonction d'informations d'habilitation desdits utilisateurs.

25 80. Procédé selon la revendication 79, caractérisé en ce que les habilitations sont traitées par groupes d'utilisateurs, deux utilisateurs d'un même groupe ayant les mêmes droits.

30 81. Procédé selon l'une des revendications 78 à 80, caractérisé en ce que les versions d'un objet sont constituées par des mises à jours successives dudit objet au niveau de postes clients de l'environnement informatique, et en ce que l'étape de création d'une nouvelle clé lors de la création d'une nouvelle version de l'objet est effectuée au niveau du poste client, une étape subséquente de transmission de la nouvelle clé à un serveur de clés étant prévue.

35 82. Procédé selon l'une des revendications 78 à 81, caractérisé en ce que les versions d'un objet sont constituées par des versions à contenus différents dudit objet, coexistant simultanément et réservées à des groupes différents d'utilisateurs, et en ce qu'une communication d'un tel objet par un utilisateur d'un premier groupe à un utilisateur d'un deuxième groupe comprend :

- 40 - la fourniture à l'utilisateur du deuxième groupe, par tout canal approprié, d'un objet multi-versions contenant le contenu de l'objet réservé au deuxième groupe, à l'état chiffré, et
- la fourniture indépendante, par un serveur de clés, à l'utilisateur du deuxième groupe, d'une clé de déchiffrement permettant le déchiffrement dudit contenu de l'objet réservé au deuxième groupe.

45 83. Procédé selon la revendication 82, caractérisé en ce qu'il comprend une étape consistant à découper l'objet multi-versions en parties homogènes du point de vue des droits d'accès, chaque partie contenant des informations de positionnement permettant son remplacement dans la structure, de telle sorte que des manipulations puissent être effectuées sur l'objet multi-versions même en l'absence de droit d'accès à la totalité de ses versions.

50 84. Procédé pour rendre accessible à différents groupes d'utilisateurs (ERC) au sein d'un environnement informatique une ressource contenant une pluralité d'objets susceptibles d'être modifiés par les utilisateurs pour obtenir différentes versions desdits objets, caractérisé en ce

117
que chaque objet est susceptible d'exister en différentes versions correspondant à des groupes d'utilisateurs différents et descendantes d'un objet multi-versions les englobant, et en ce qu'il comprend, lors de l'accès par un utilisateur à un objet, la présentation à celui-ci des versions de l'objet correspondant aux groupes auxquels il appartient, chaque version d'objet étant obtenue par transformation à partir d'une autre version dudit objet.

85. Procédé selon la revendication 84, caractérisé en ce qu'il est mis en œuvre à l'aide d'un procédé de gestion selon la revendication 62, des transformations prédéterminées spécifiques permettant d'obtenir chacune des versions de l'objet.

86. Procédé selon la revendication 85, caractérisé en ce qu'une action de transclusion par un utilisateur d'un objet appartenant à un objet multi-versions provoque une transclusion de l'ensemble de l'objet multi-versions.

87. Procédé selon l'une des revendications 84 à 86, caractérisé en ce que chaque objet est également susceptible d'exister au sein d'un même groupe d'utilisateurs dans différentes versions correspondant à différents utilisateurs, ces versions pouvant être également présentées lors de l'accès à l'objet par les utilisateurs dudit groupe.

88. Procédé selon la revendication 87, caractérisé en ce qu'il comprend en outre la mise en œuvre de moyens de calcul de scores de pertinences des différentes versions correspondant aux différents groupes et/ou aux différents utilisateurs pour présenter sélectivement les versions les plus pertinentes.

25



Fig. 1-a

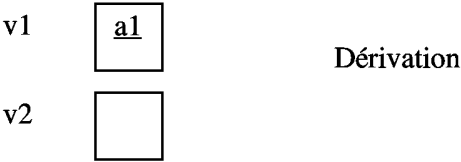


Fig. 1-b

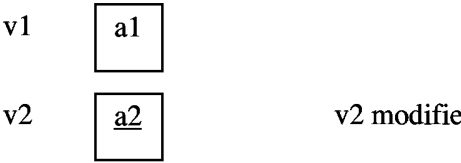


Fig. 1-c

branche 1 branche 2

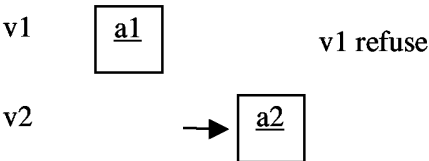


Fig. 1-d

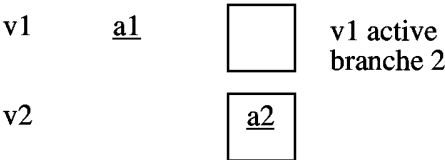


Fig. 1-e

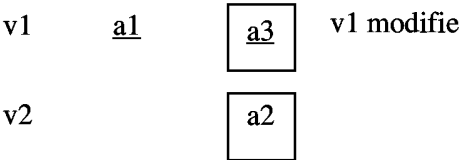


Fig. 1-f

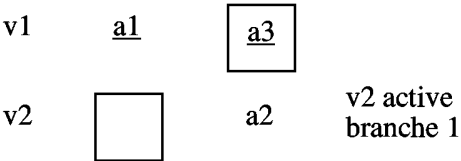


Fig. 1-g

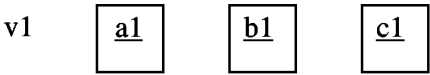


Fig. 2-a

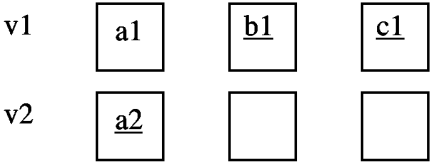


Fig. 2-b

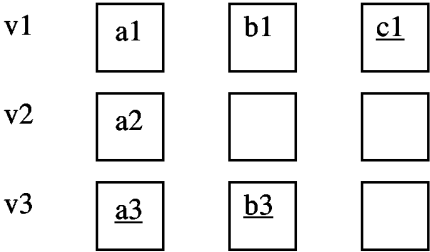


Fig. 2-c

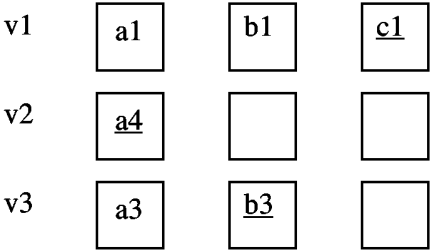


Fig. 2-d

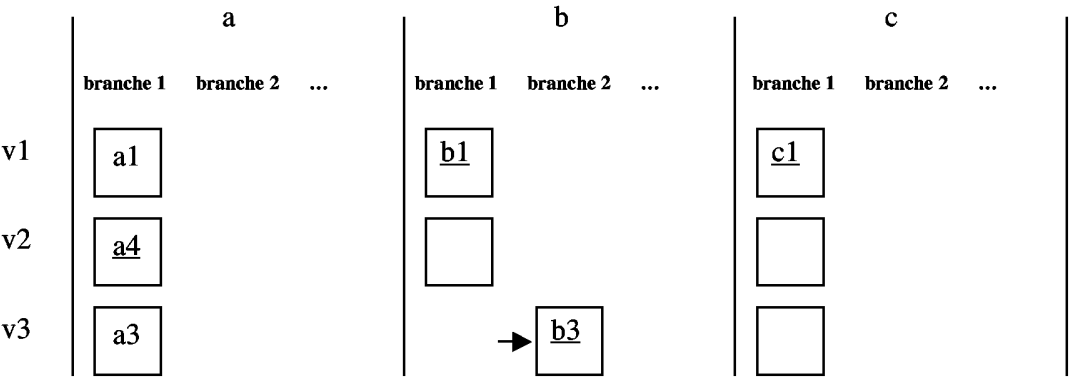


Fig. 3-a

5

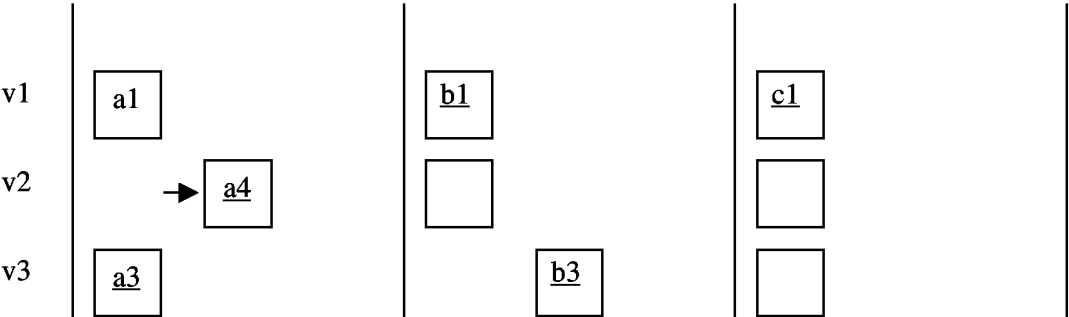


Fig. 3-b

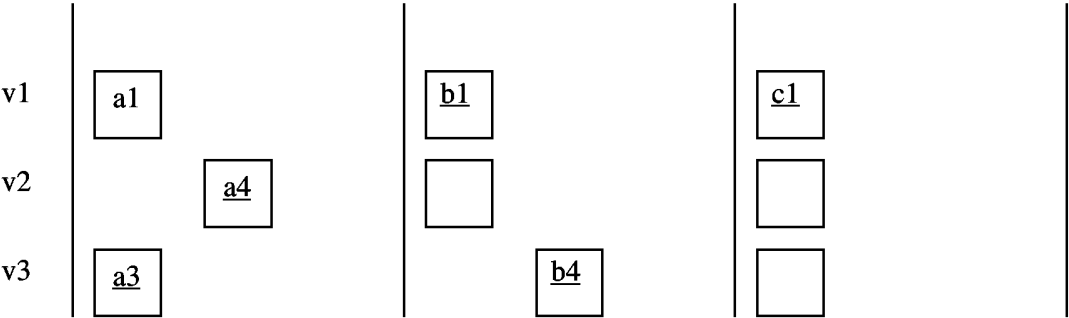


Fig. 3-c

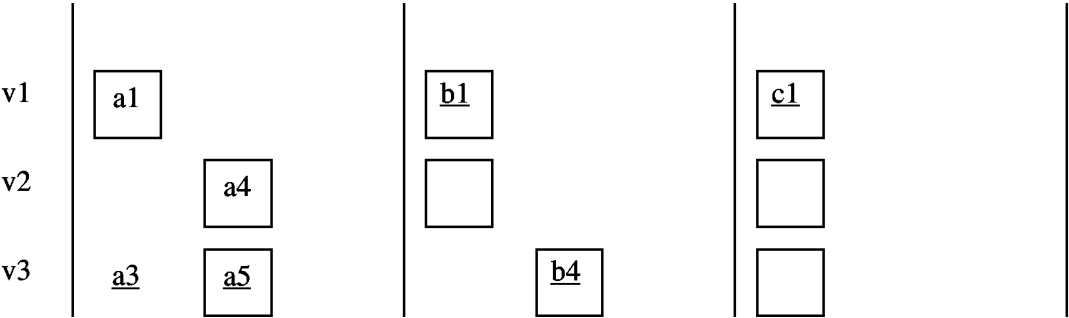


Fig. 3-d

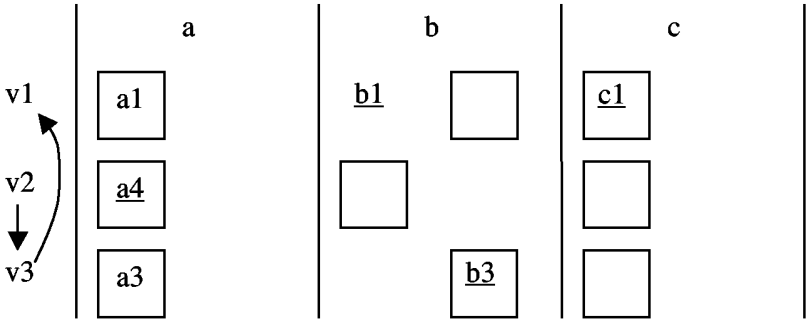


Fig. 4-a

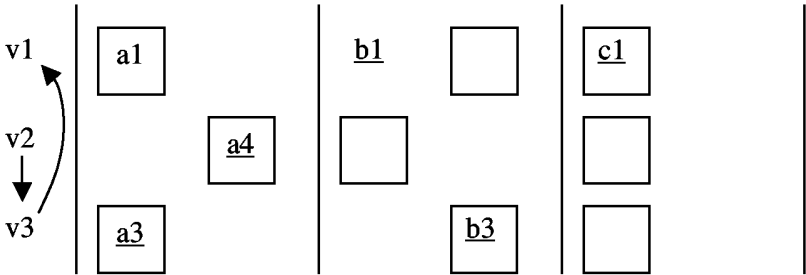


Fig. 4-b

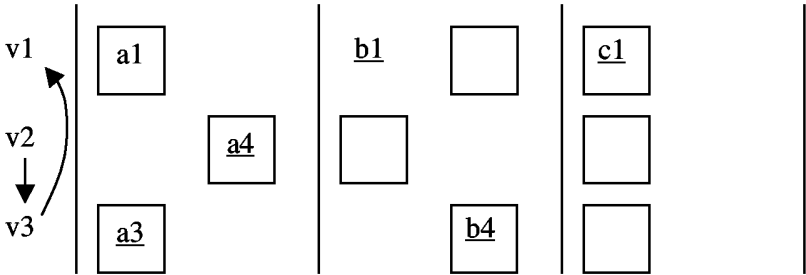


Fig. 4-c

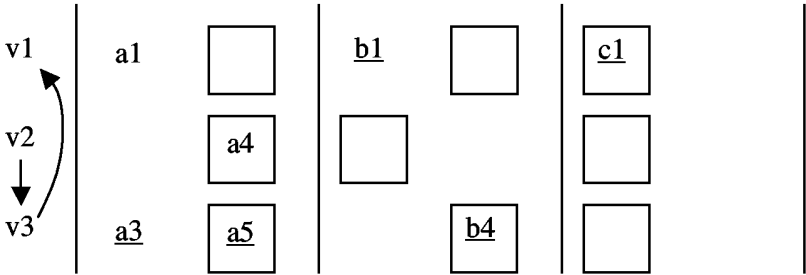


Fig. 4-d

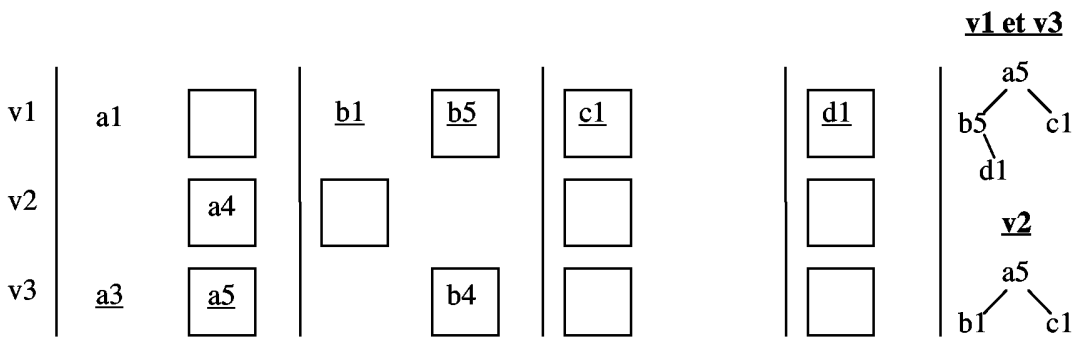
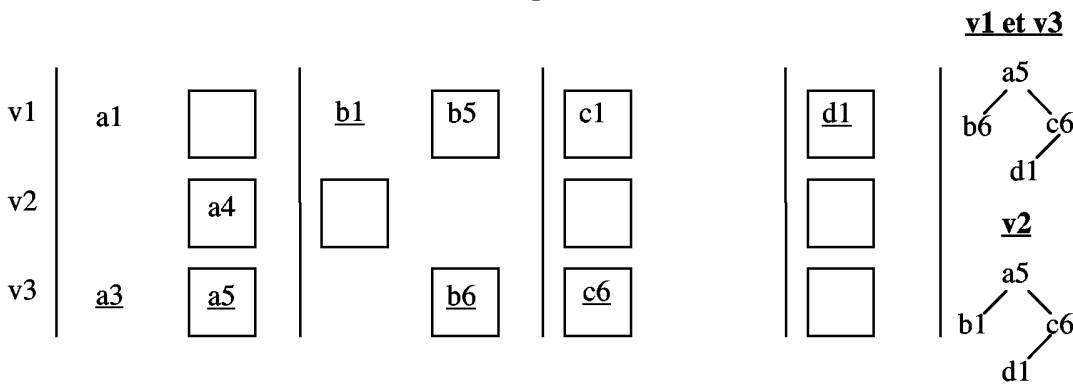


Fig. 5-a



5

Fig. 5-b

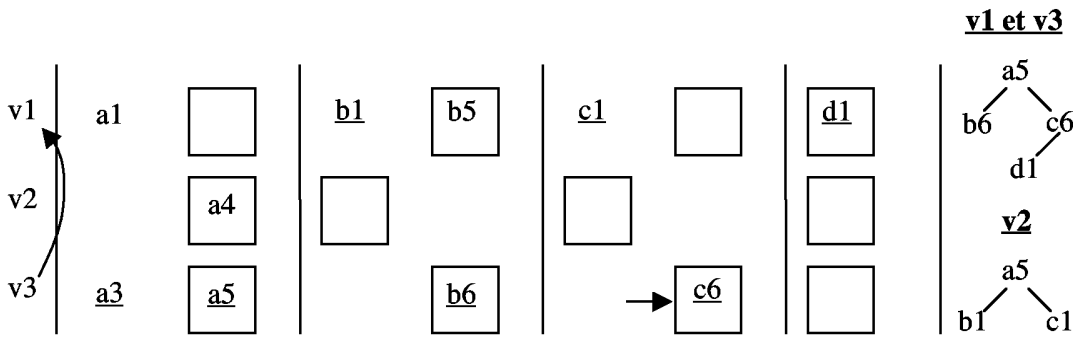
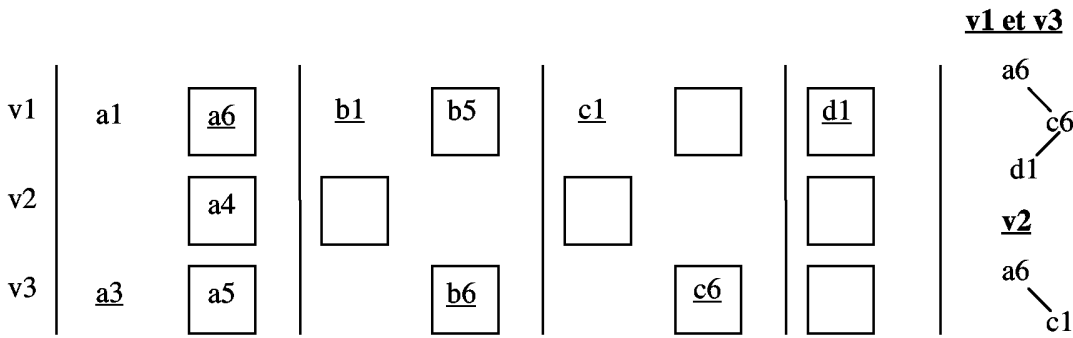


Fig. 5-c



10

Fig. 5-d

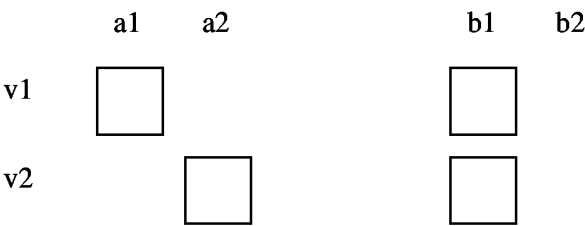


Fig. 6-a

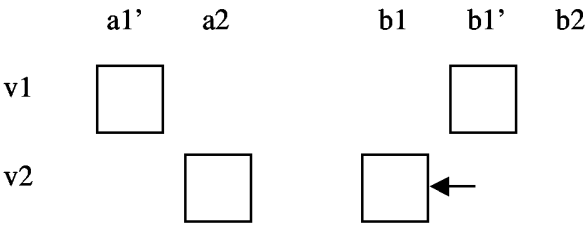


Fig. 6-b

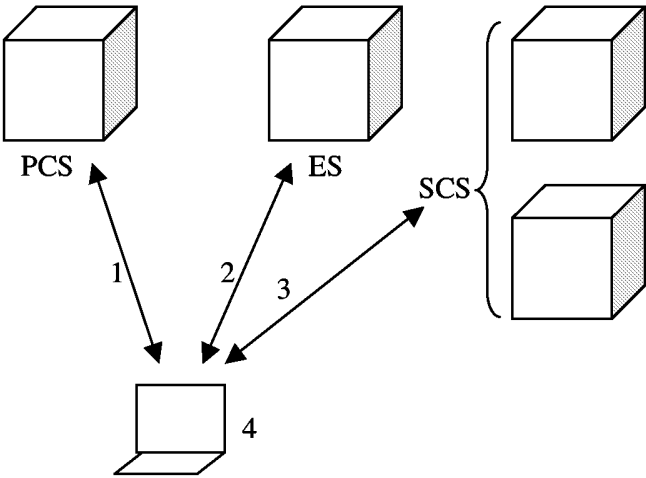


Fig. 7



Figure 8

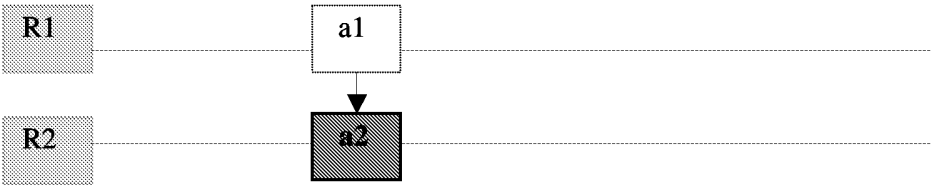


Figure 9

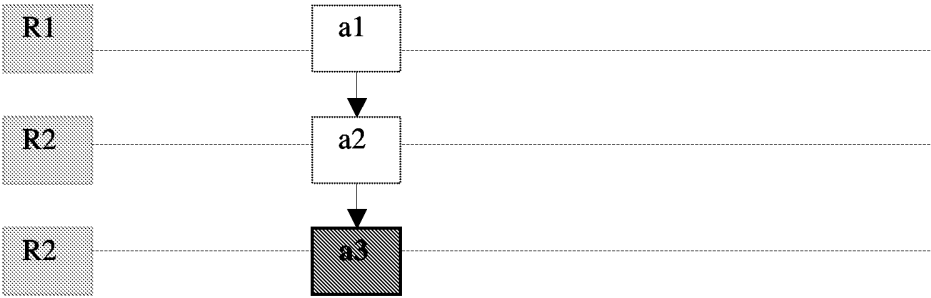


Figure 10

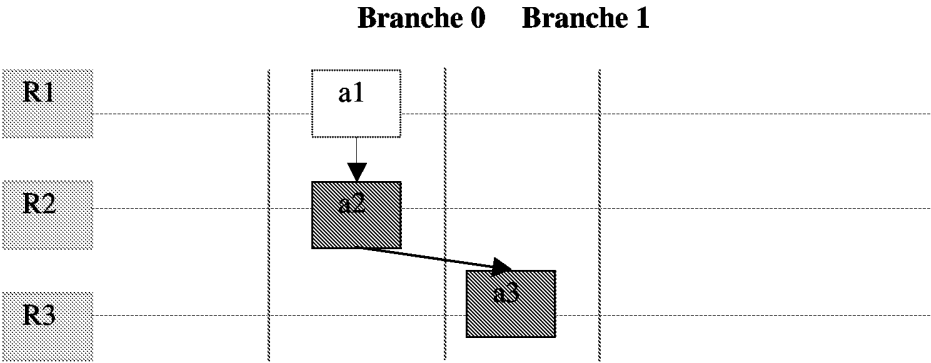


Figure 11

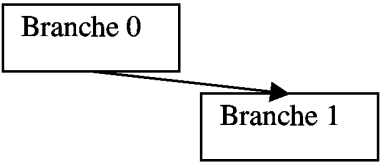


Figure 12

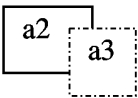


Figure 13

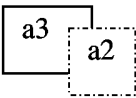
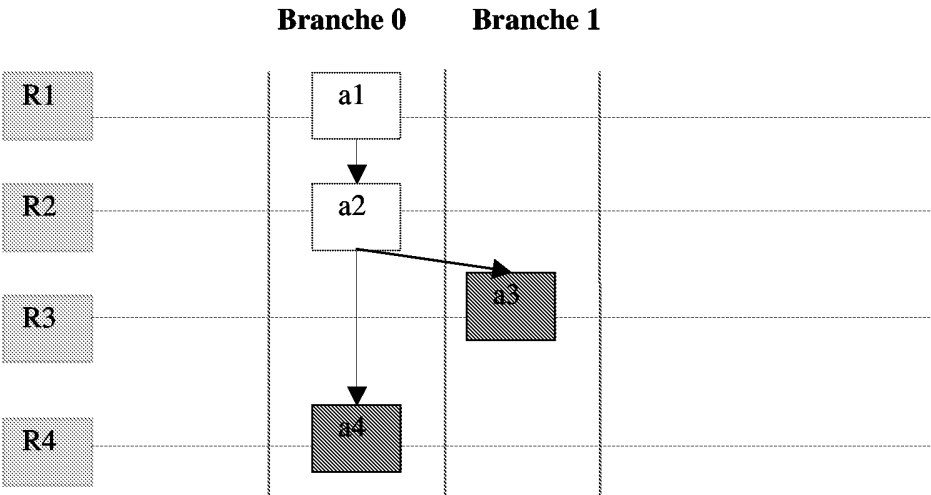


Figure 14

5



10

Figure 15

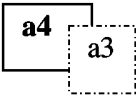


Figure 16

15

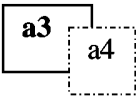
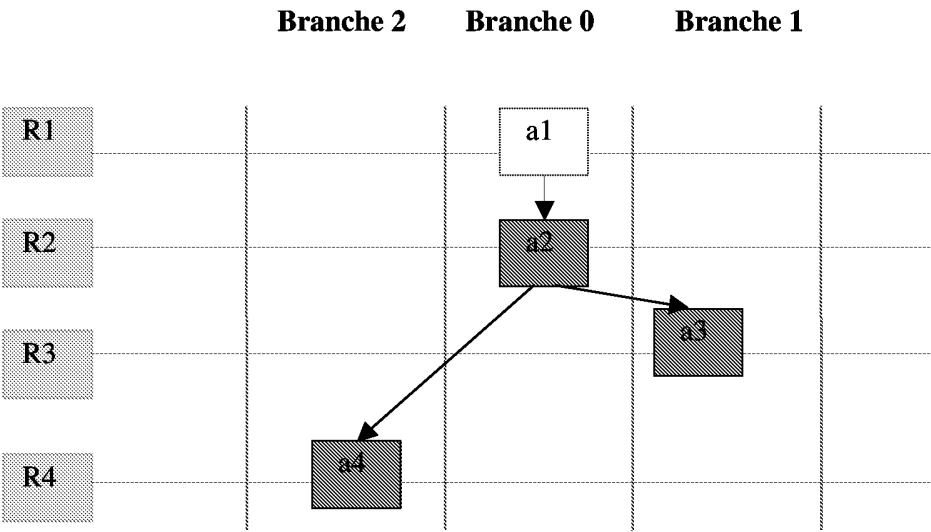


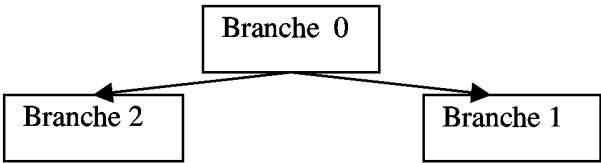
Figure 17

20



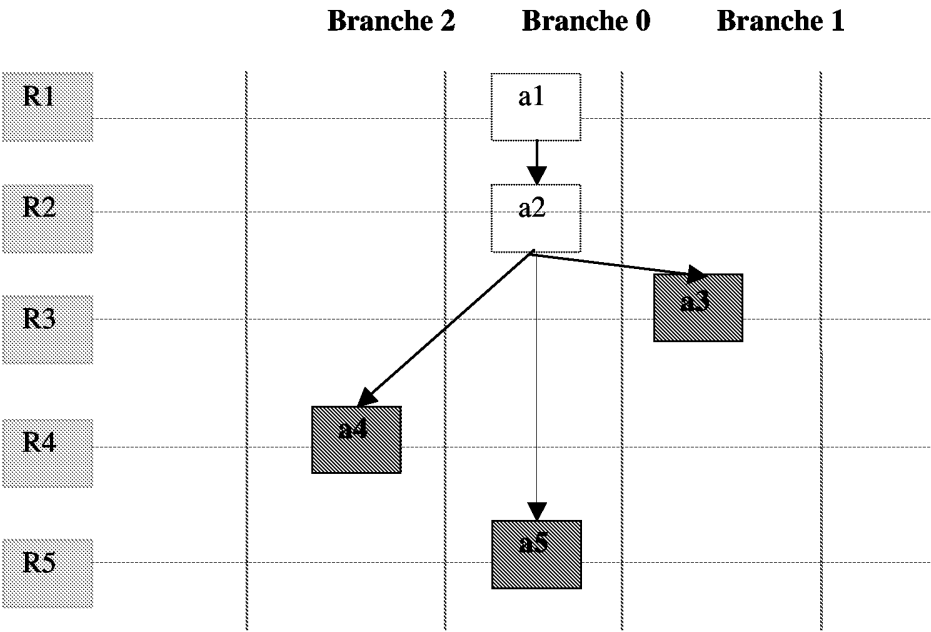
5

Figure 18



10

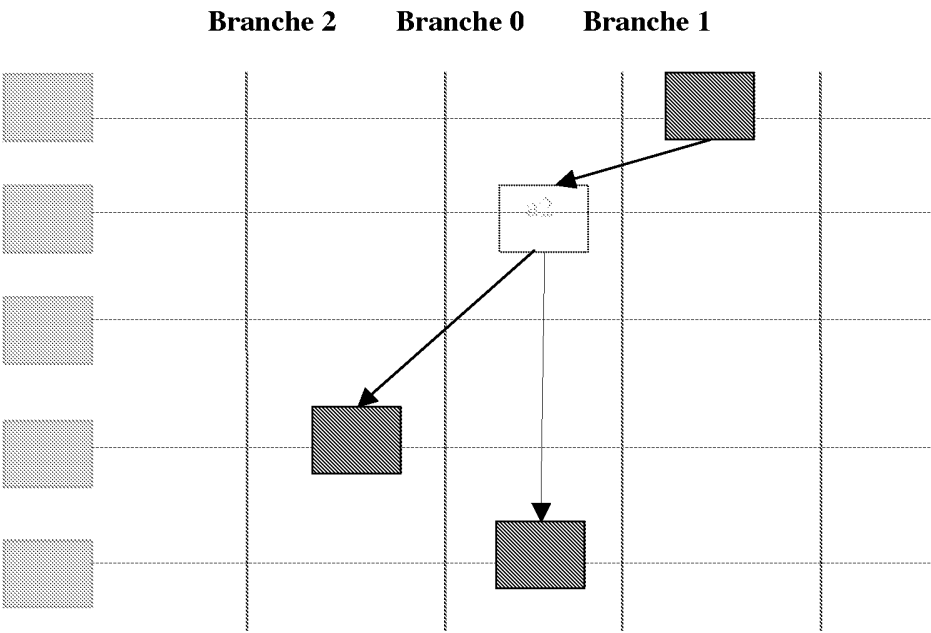
Figure 19



15

Figure 20

20



5

Figure 21

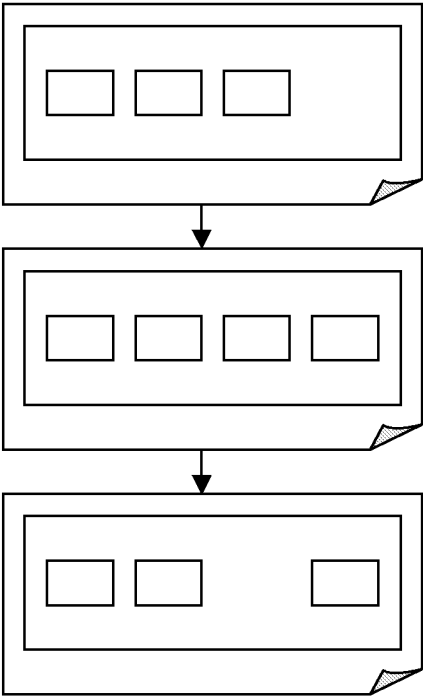
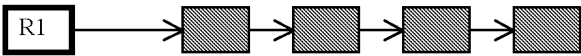


Fig. 22



10

Fig. 23

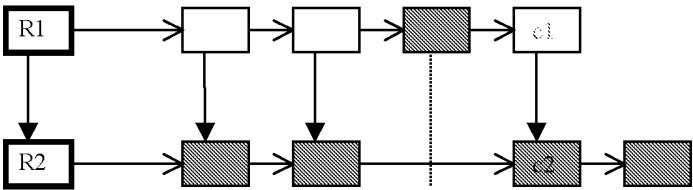


Fig. 24

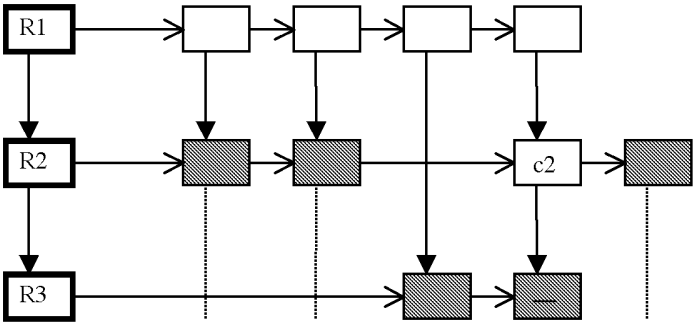


Fig. 25

5

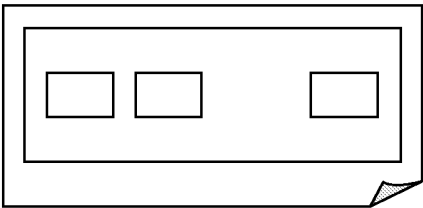


Fig. 26

10

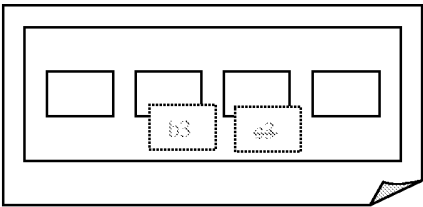


Fig. 27

15

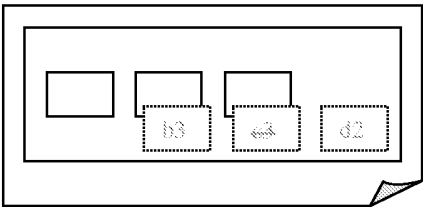


Fig. 28

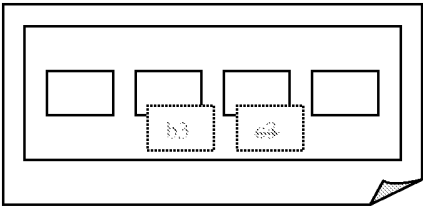


Fig. 29

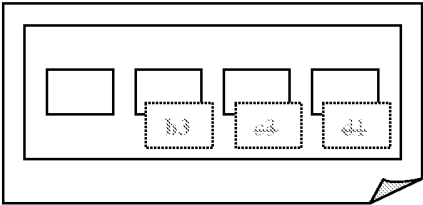


Fig. 30

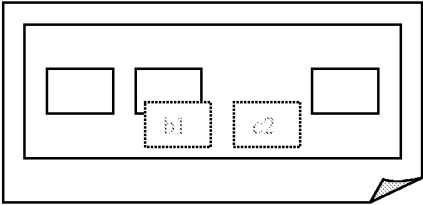


Fig. 31

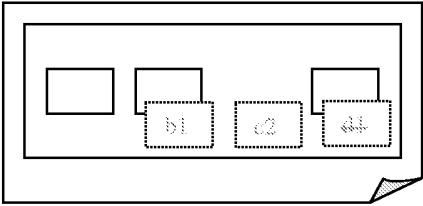


Fig. 32

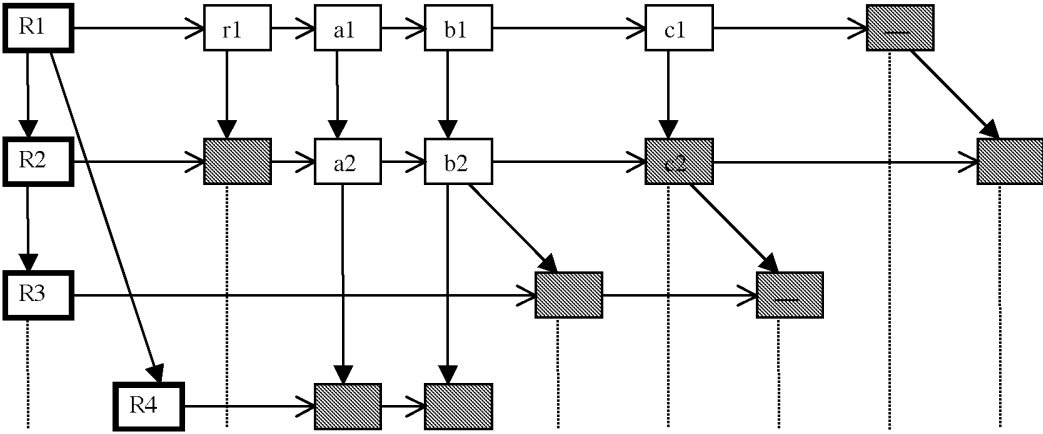


Fig. 33

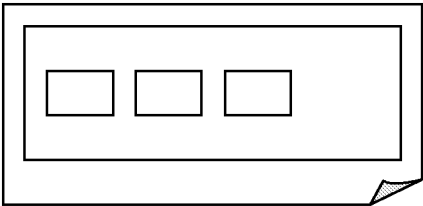


Fig. 34

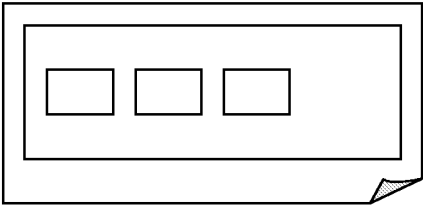


Fig. 35

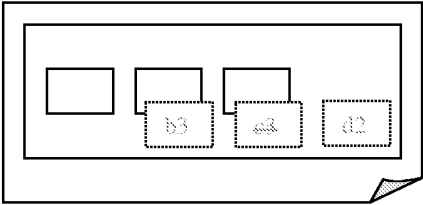


Fig. 36

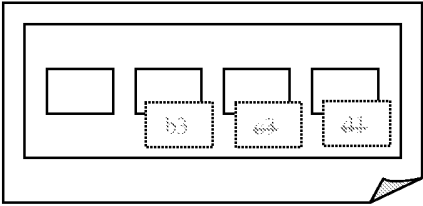


Fig. 37

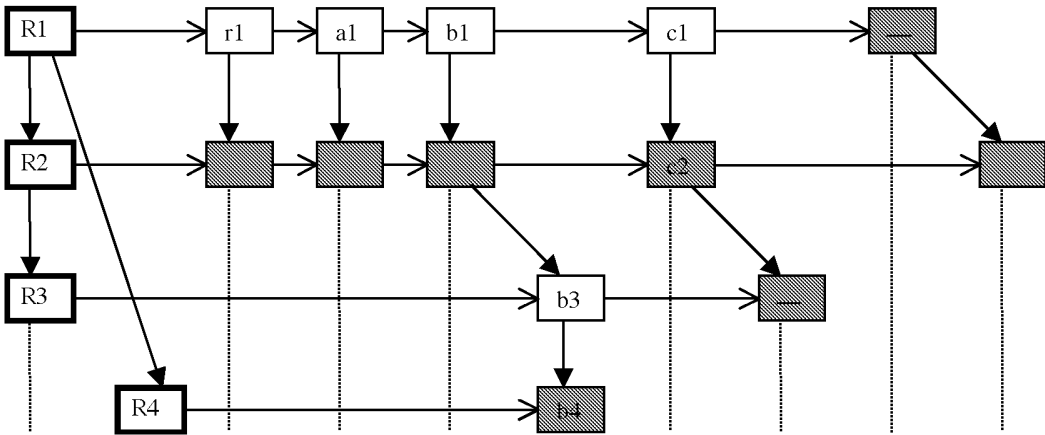


Fig. 38

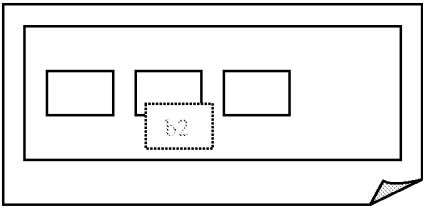


Fig. 39

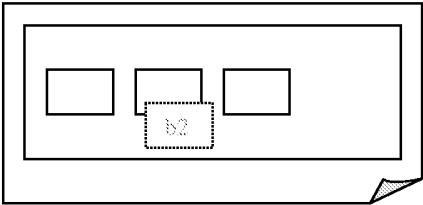


Fig. 40

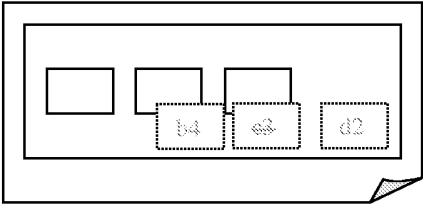


Fig. 41

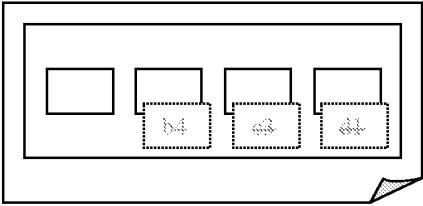


Fig. 42

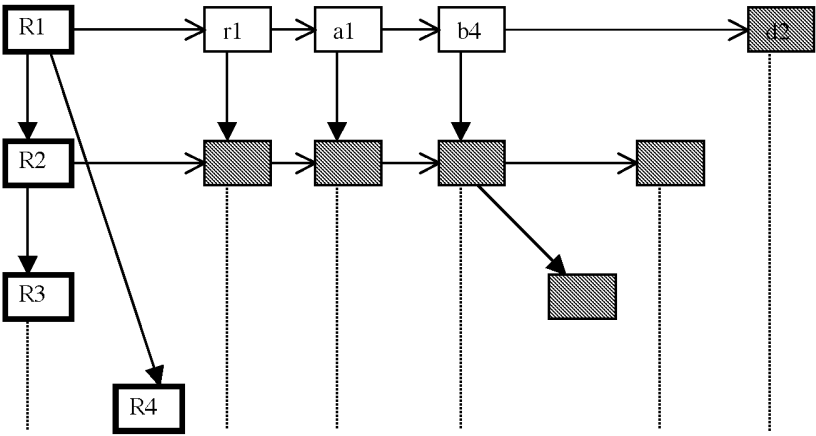


Fig. 43

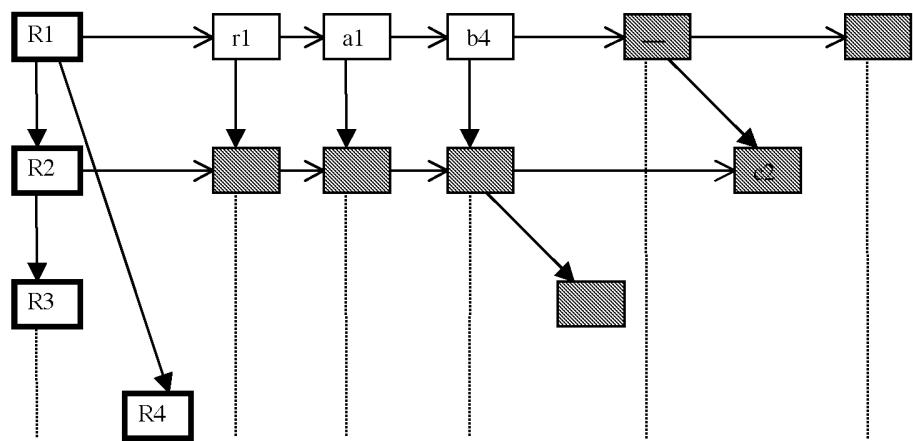


Fig. 44

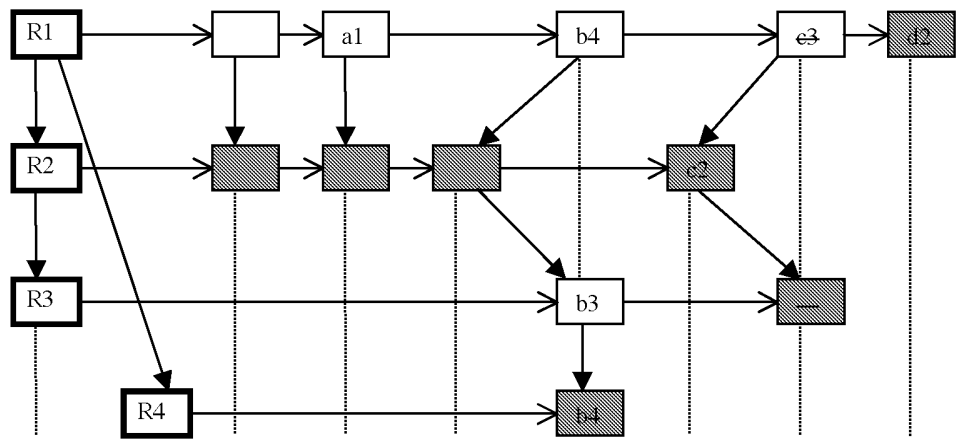


Fig. 45

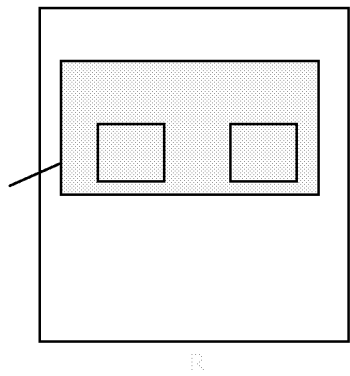


Fig. 46

5

10

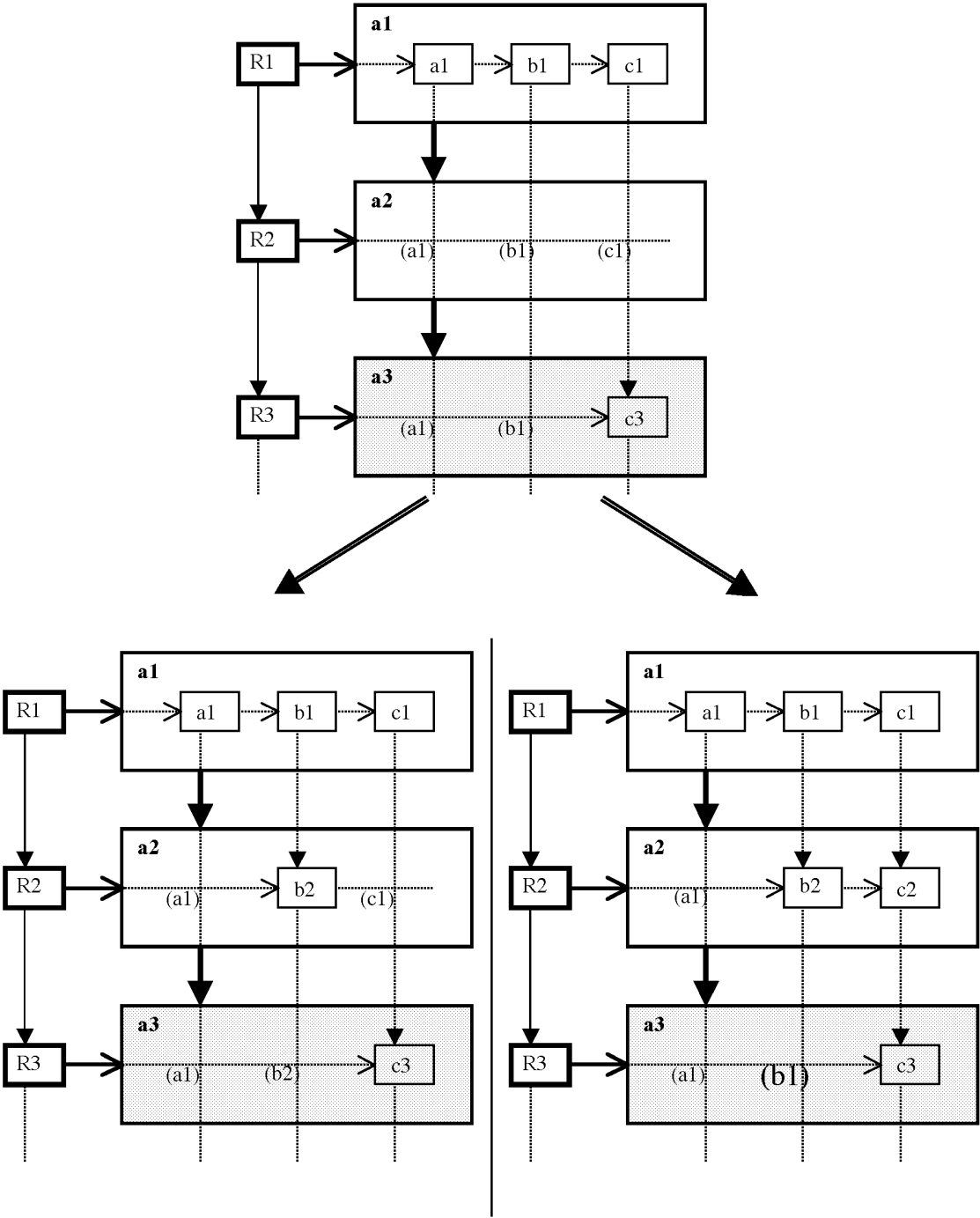


Fig. 47

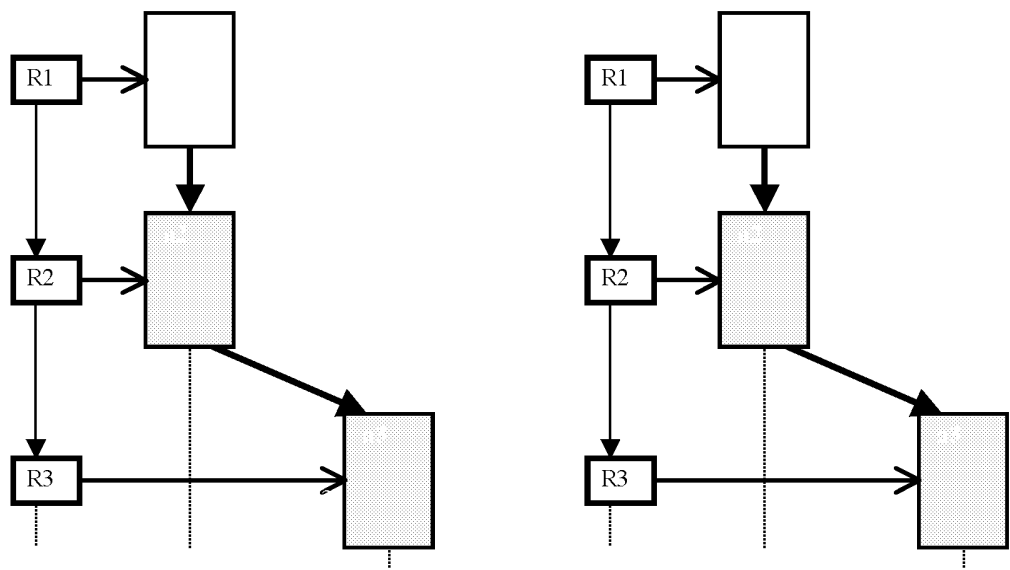


Fig. 48

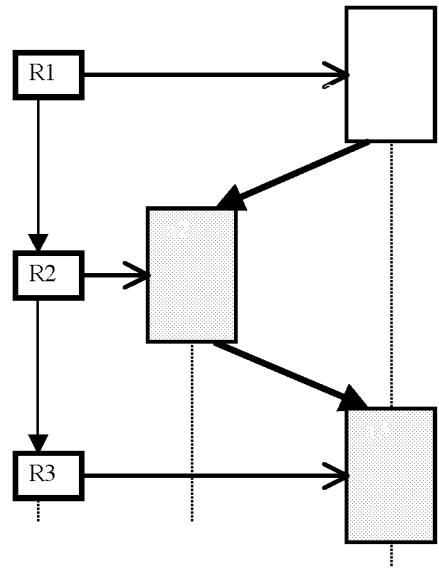
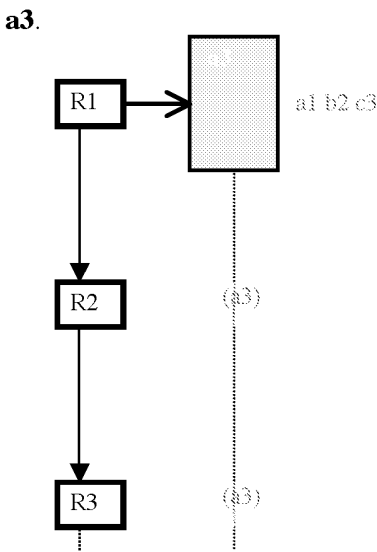


Fig. 49

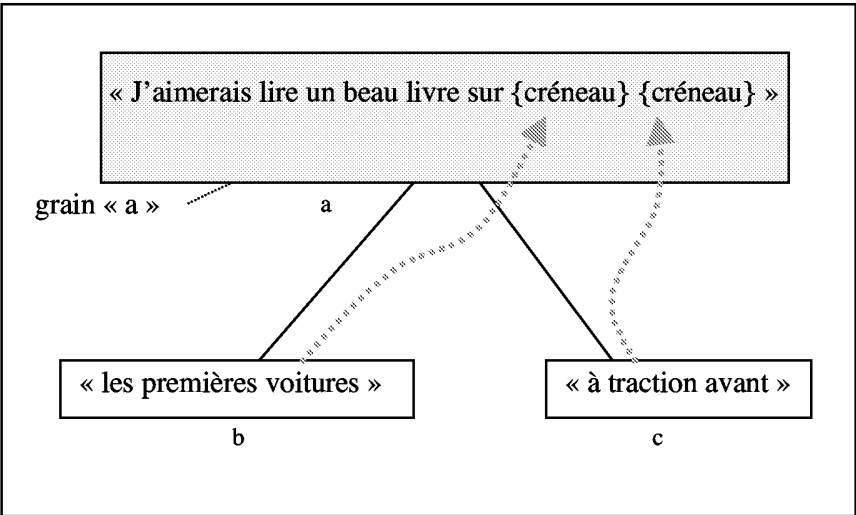


Fig. 50

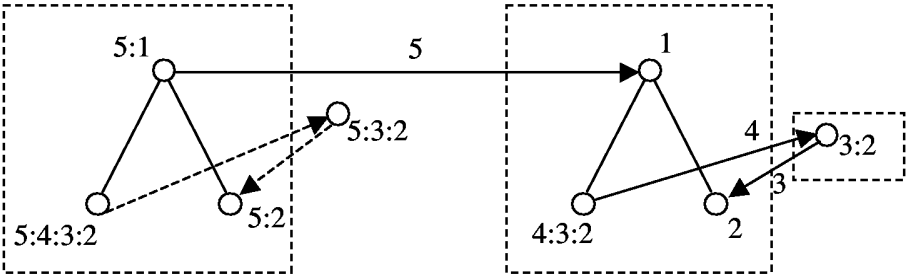


Fig. 51

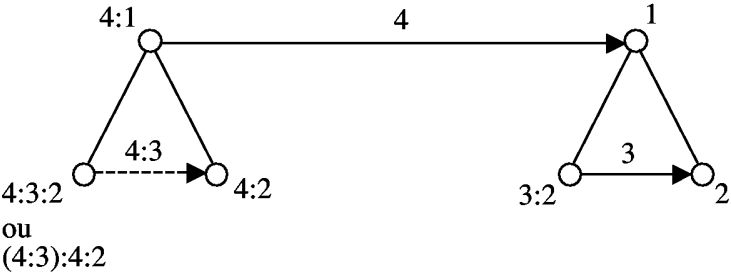


Fig. 52

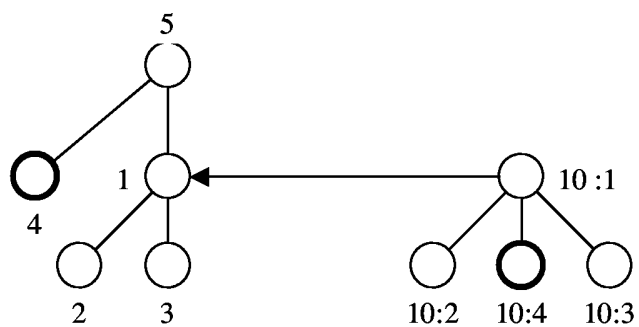
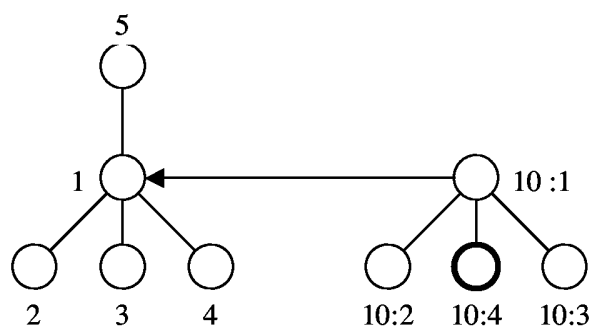


Figure 53

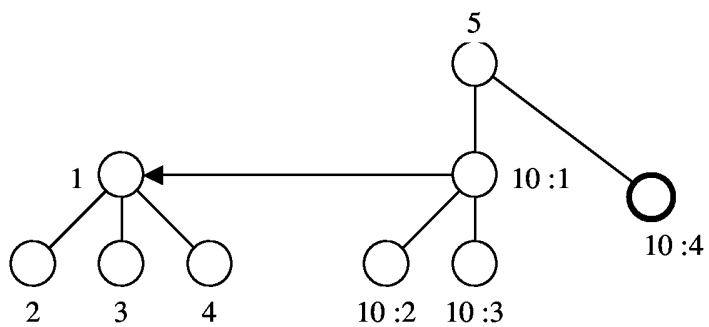
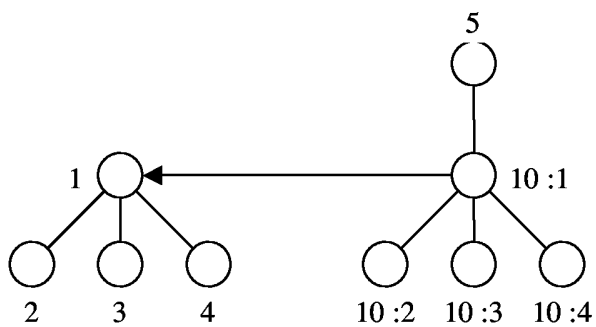


Figure 54

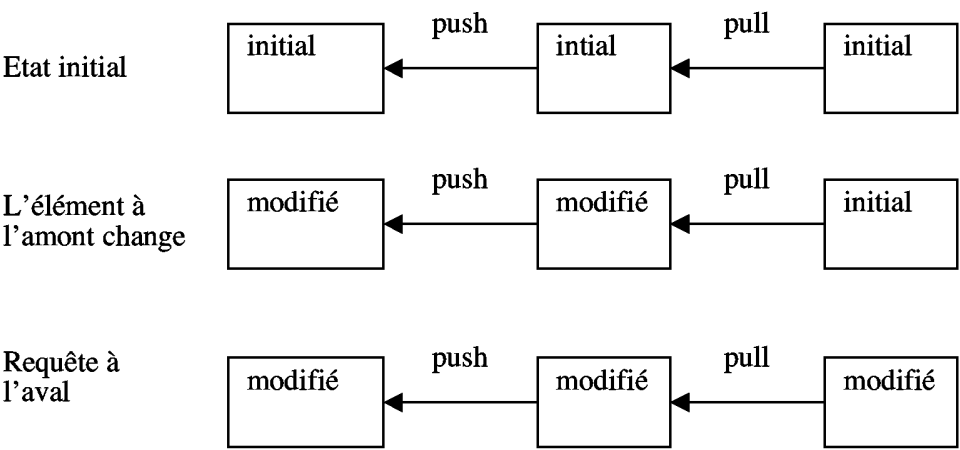


Figure 55

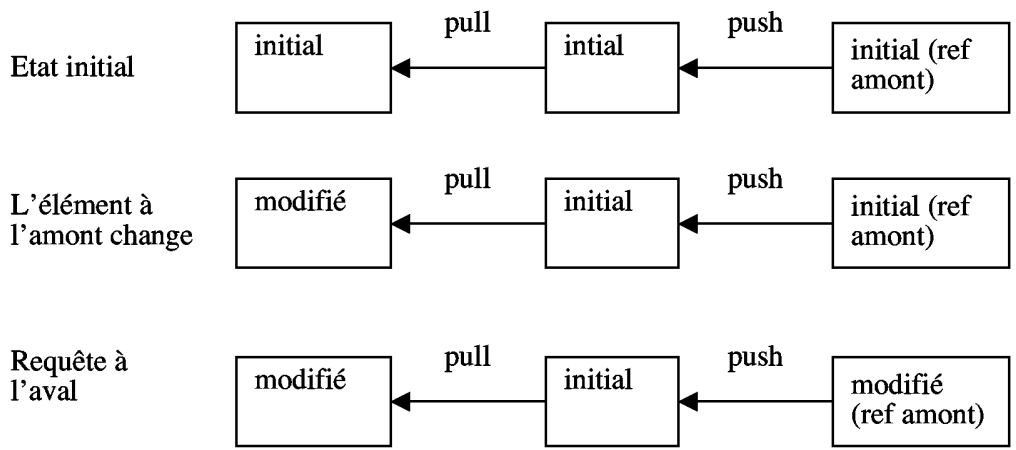


Figure 56

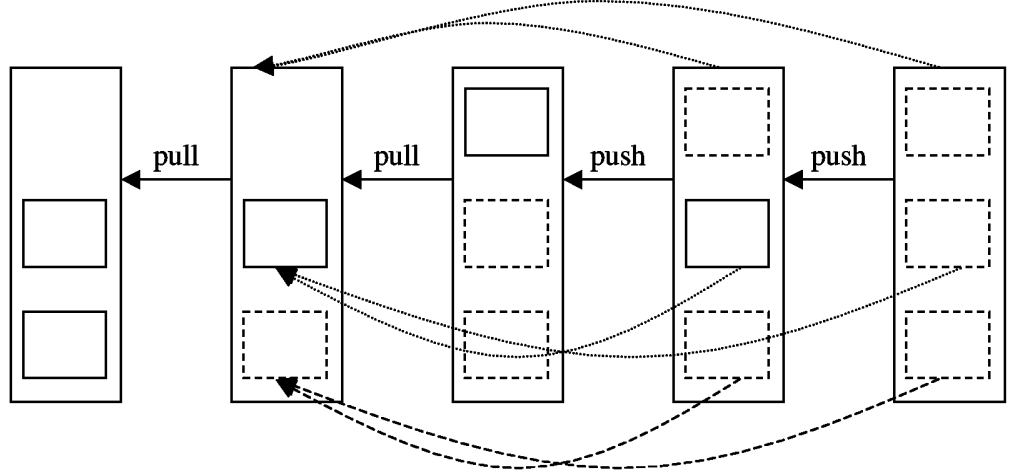


Figure 57

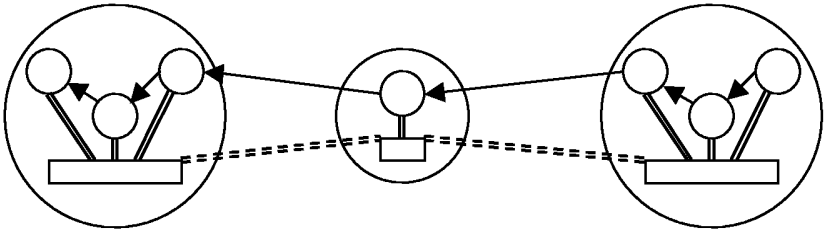


Figure 58

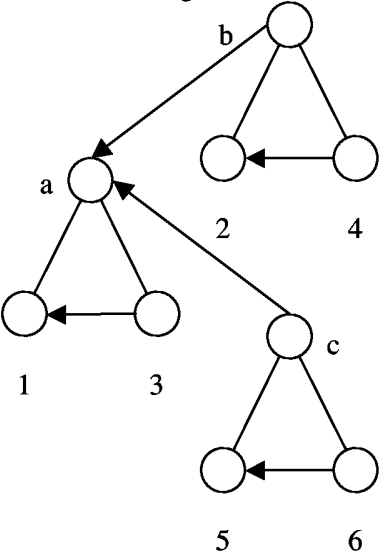


Figure 59

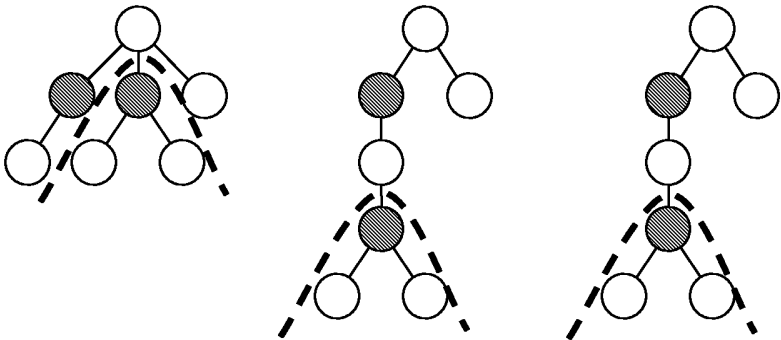


Figure 60

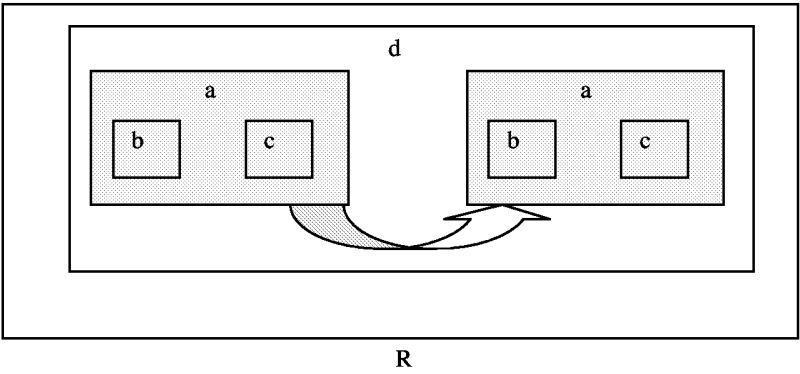


Fig. 61

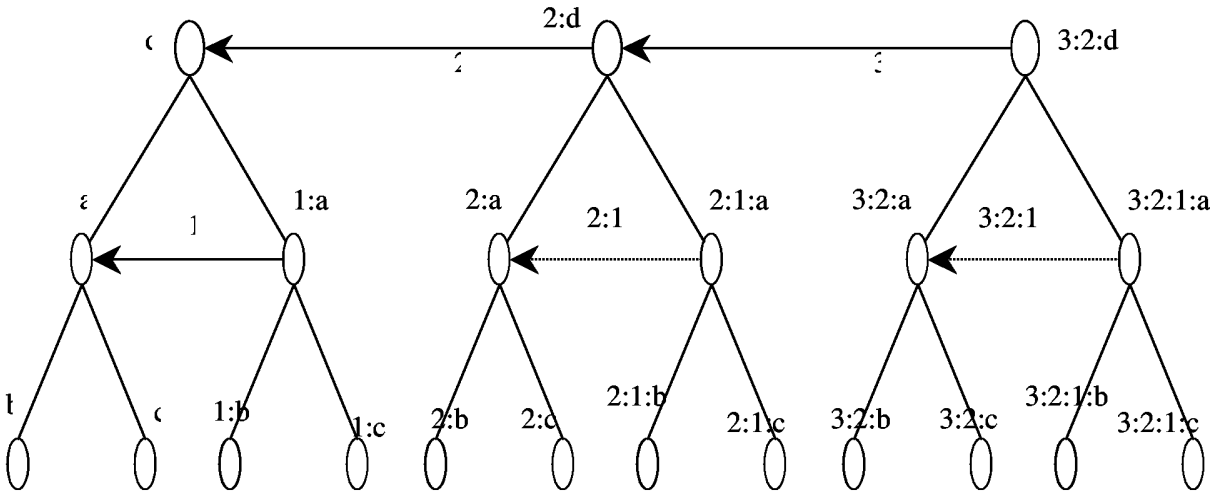


Fig. 62

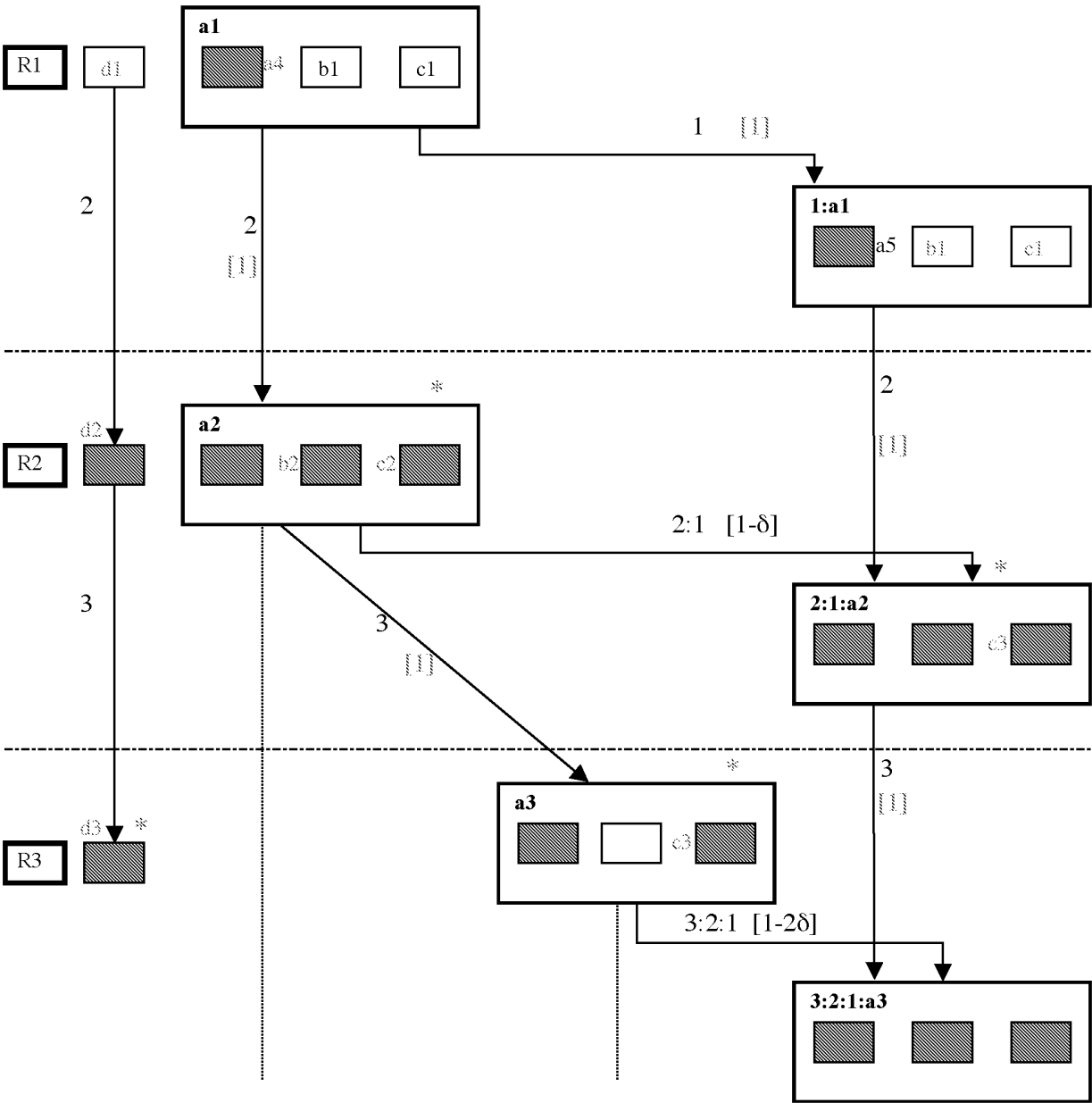
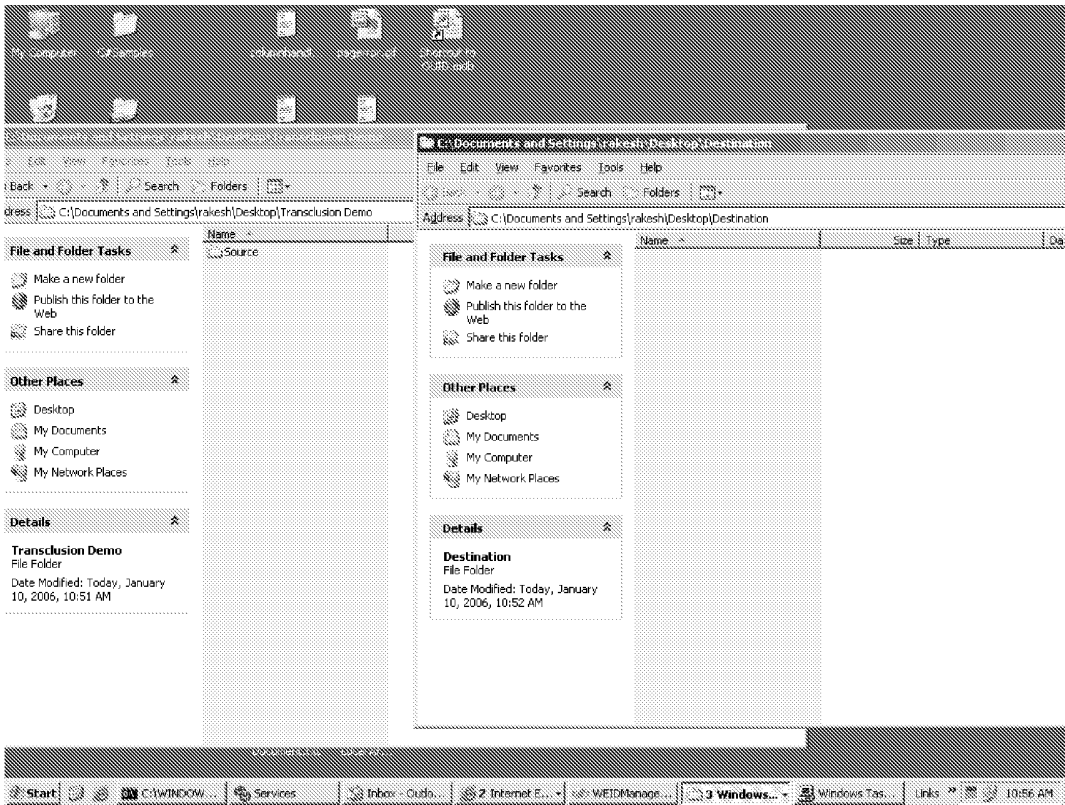


Fig. 63

Source and Destination for Transclusion



5

Figure 64

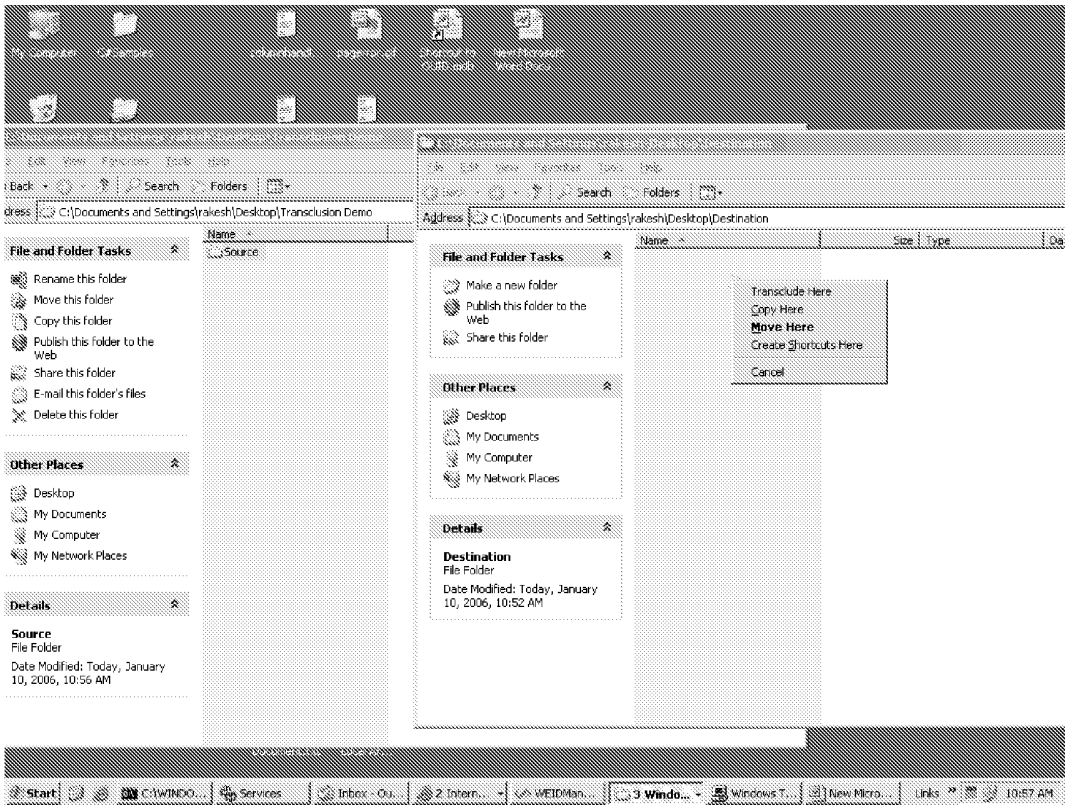
10

15

20

25

Doing drag and drop using right mouse button



5

Figure 65

5 **Choosing the ‘ Transclude Here’ Option from the drag and drop contextual menu**

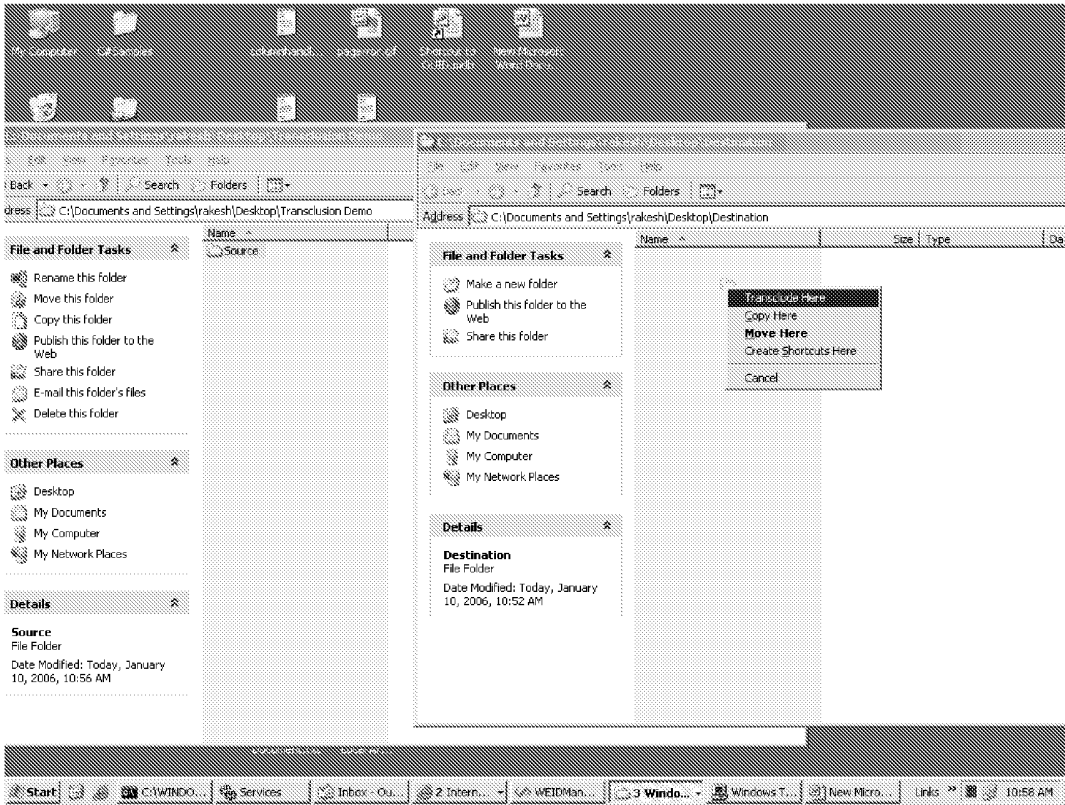
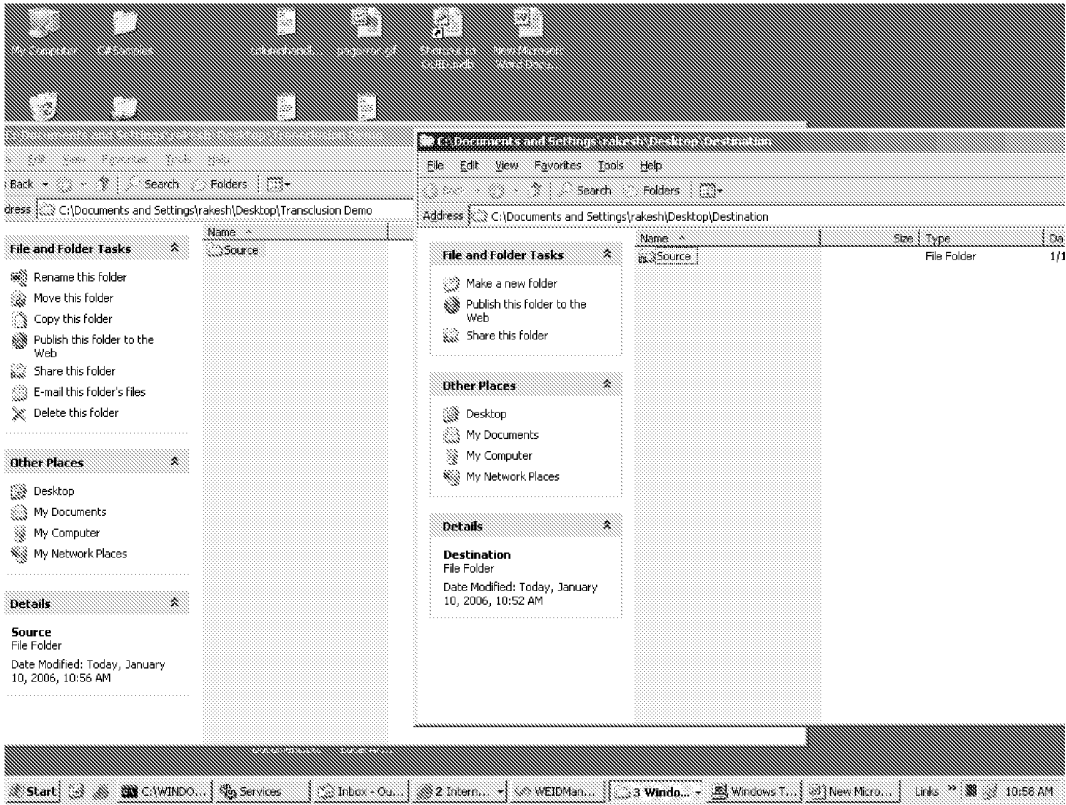


Figure 66

5

After Transclusion the ‘ Transclusion Result’ is displayed with red asterisk icon overlay



10

Figure 67

15

20

25

30

5 **Selecting the 'Transclusion Source' column from the available columns**

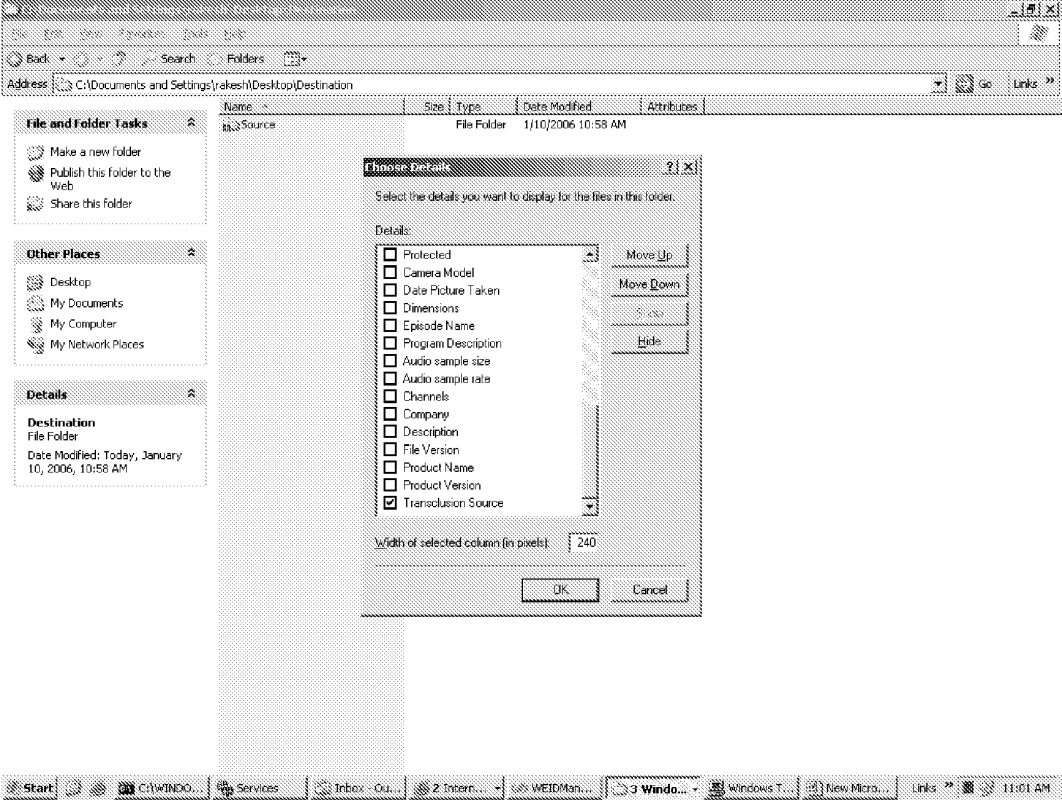


Figure 68

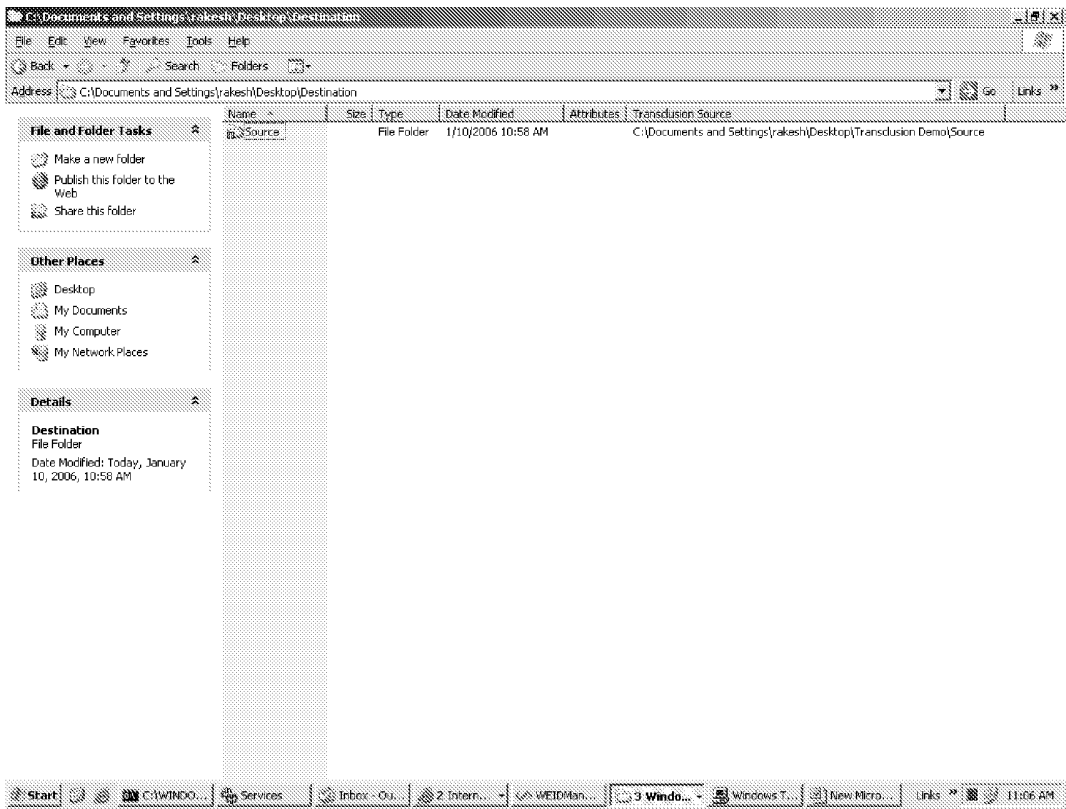
10

15

20

25

Transclusion Source column being displayed



5

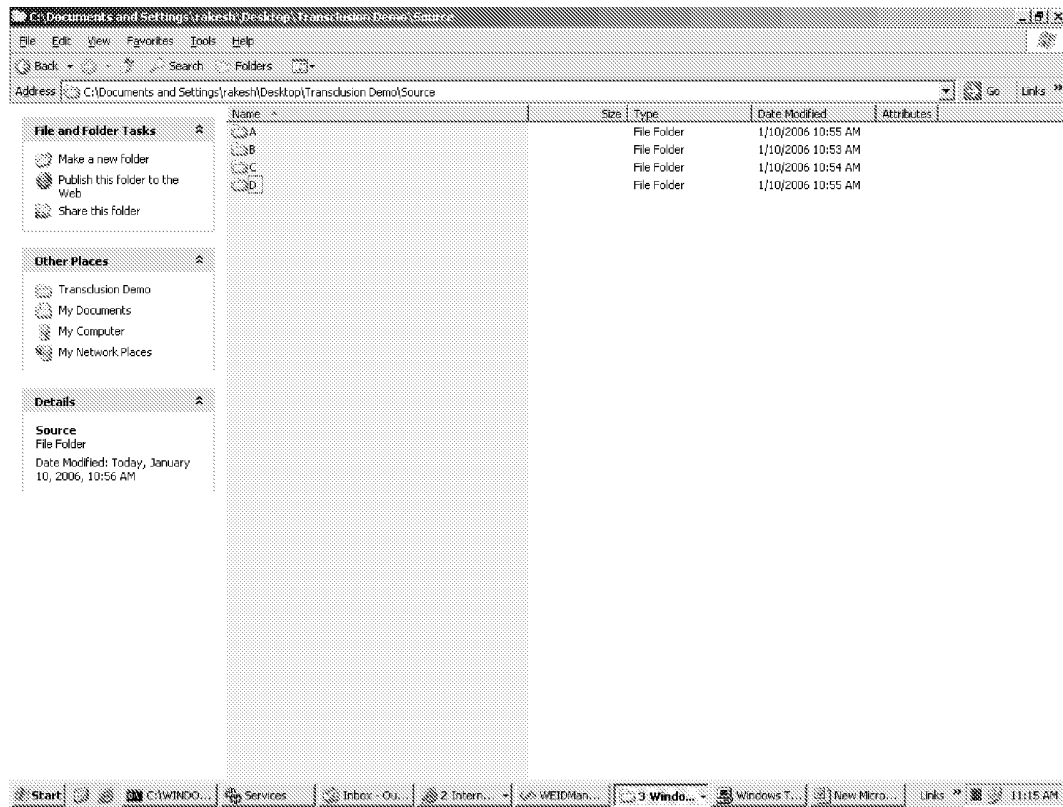
Figure 69

10

15

20

25

View of Source directory before any modification

5 Corresponding view at the result side before any modification

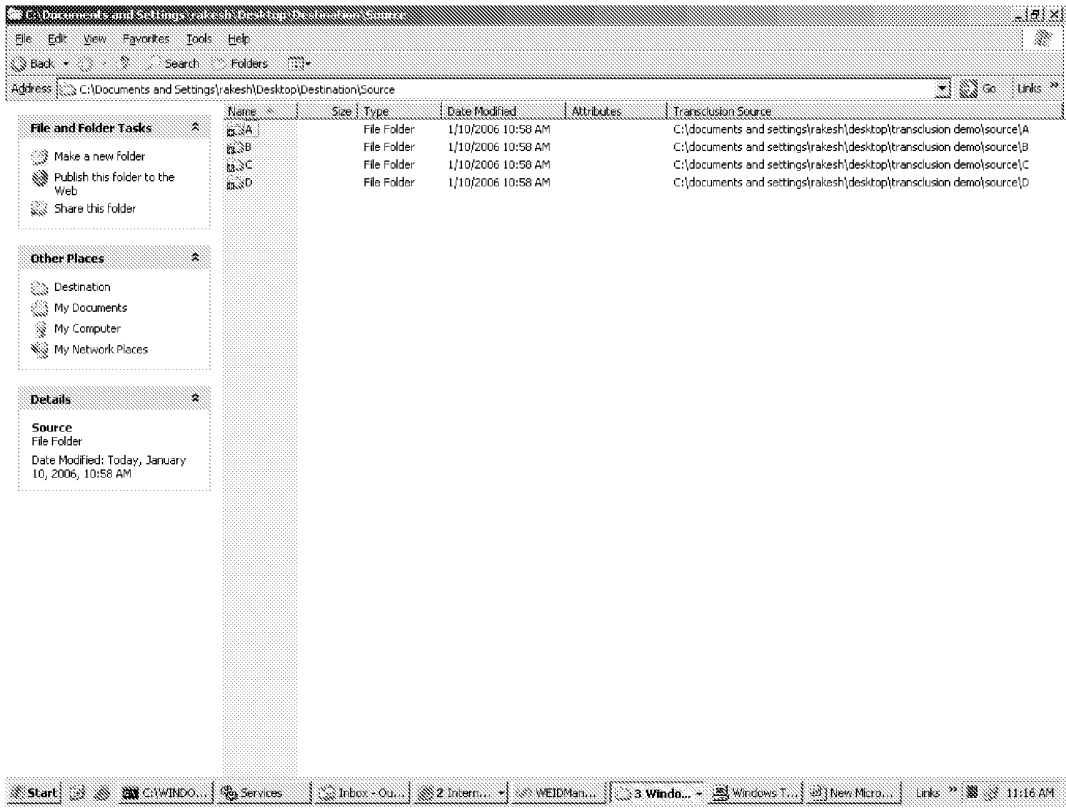


Figure 71

5

Updates: Modifying the source side

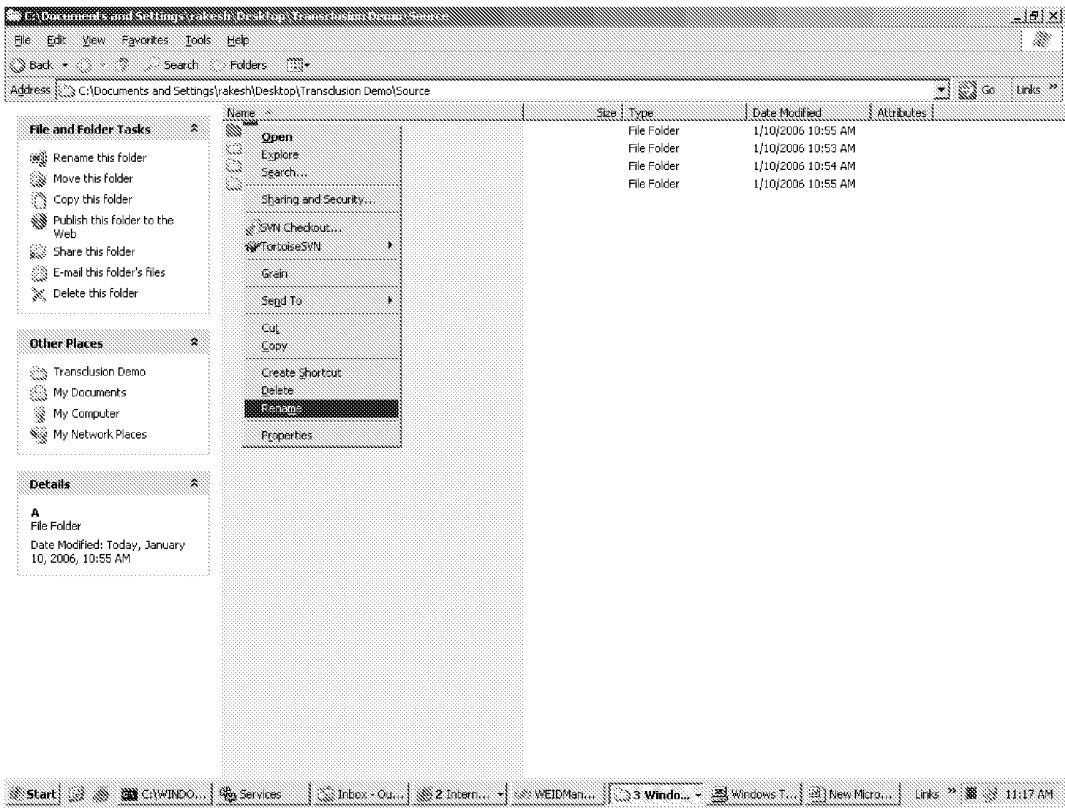


Figure 72

10

15

20

Updates: After modification of source, view at source side

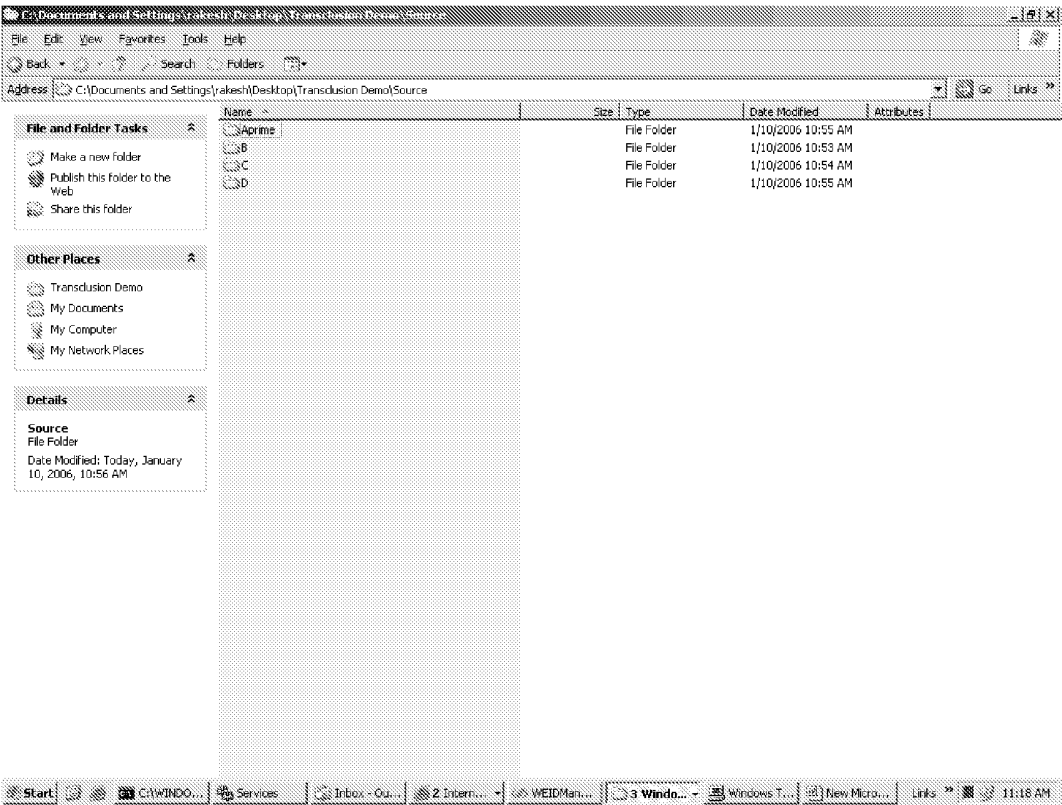


Figure 73

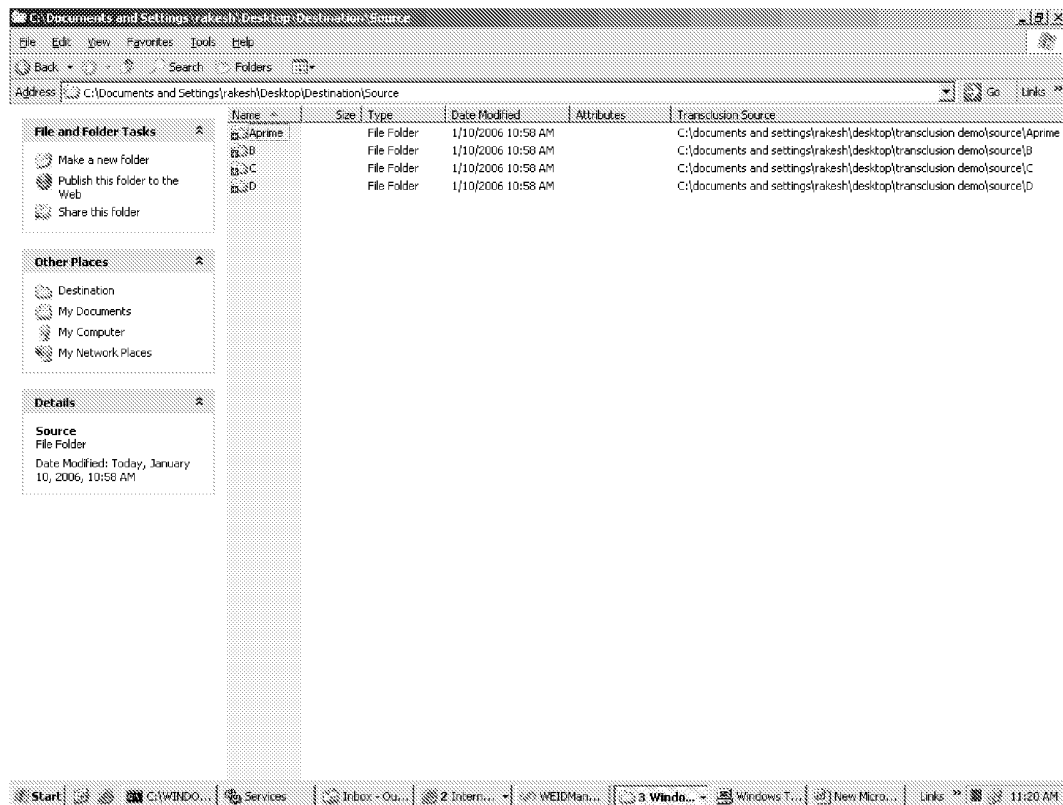
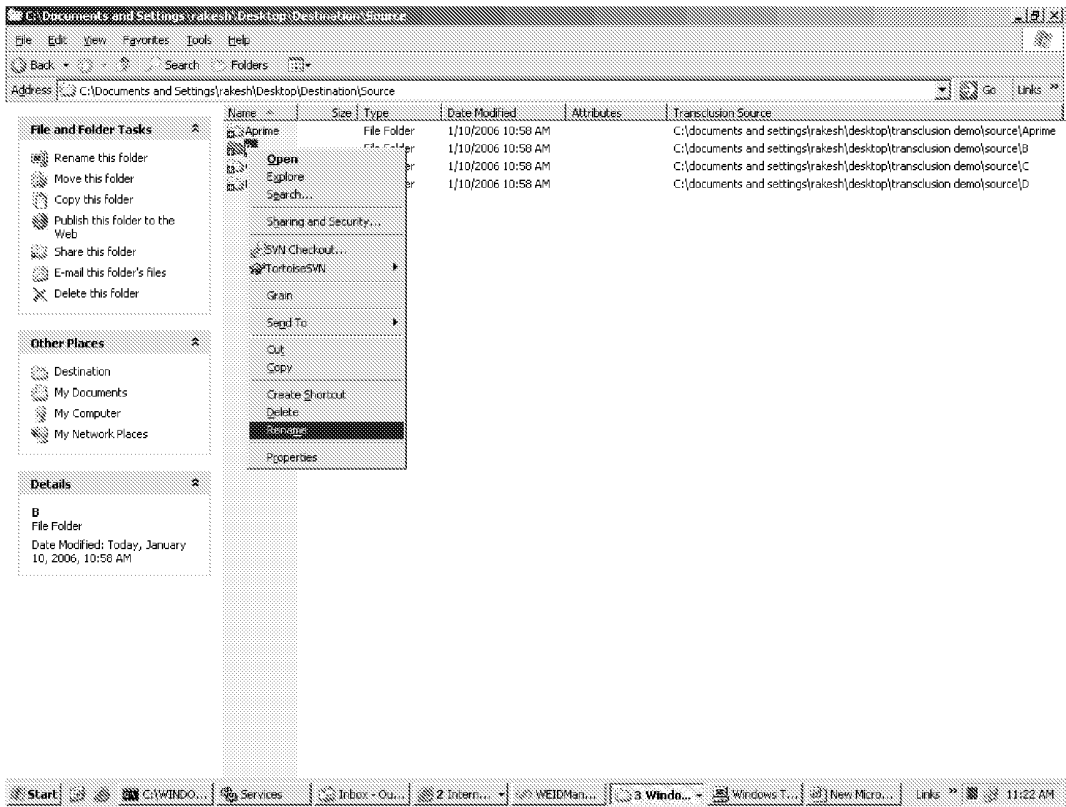
Updates: After modification of source, view at result side

Figure 74

Contributions: modifying the result side



5

Figure 75

10

Contributions: After modification of result, view at result side
Black tick icon overlay representing that result is being modified and also the entry in the Transclusion column displaying ‘Locally Modified’ status

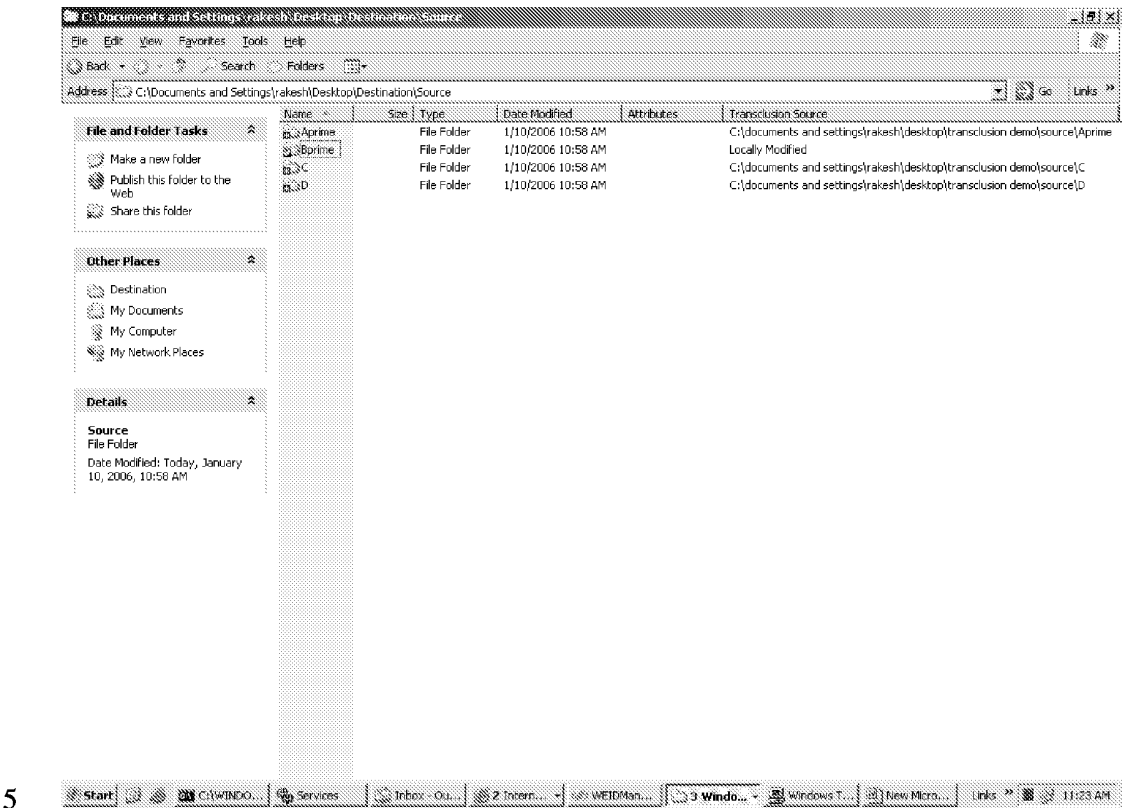


Figure 76

Contributions: After modification of result, view at source side
The blue plus icon overlay representing that B has got contributions

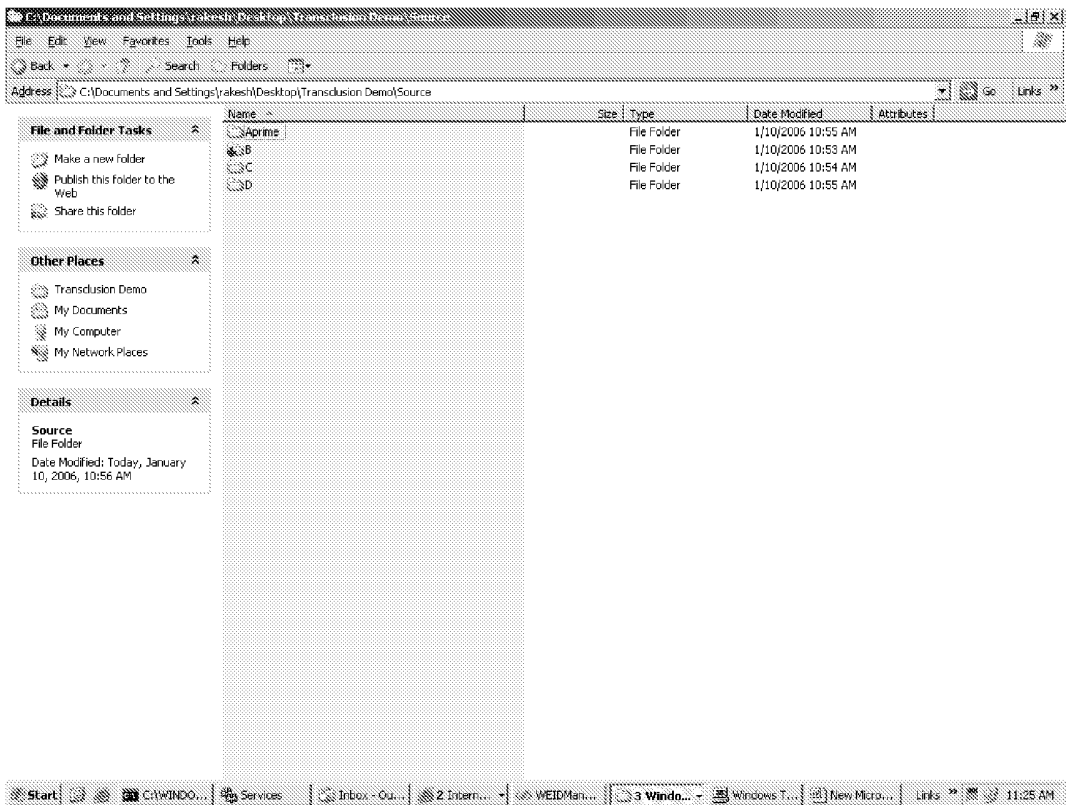


Figure 77

Contributions: After modification of result, view at source side
The right click contextual menu shows the ‘View Contribution’ menu item

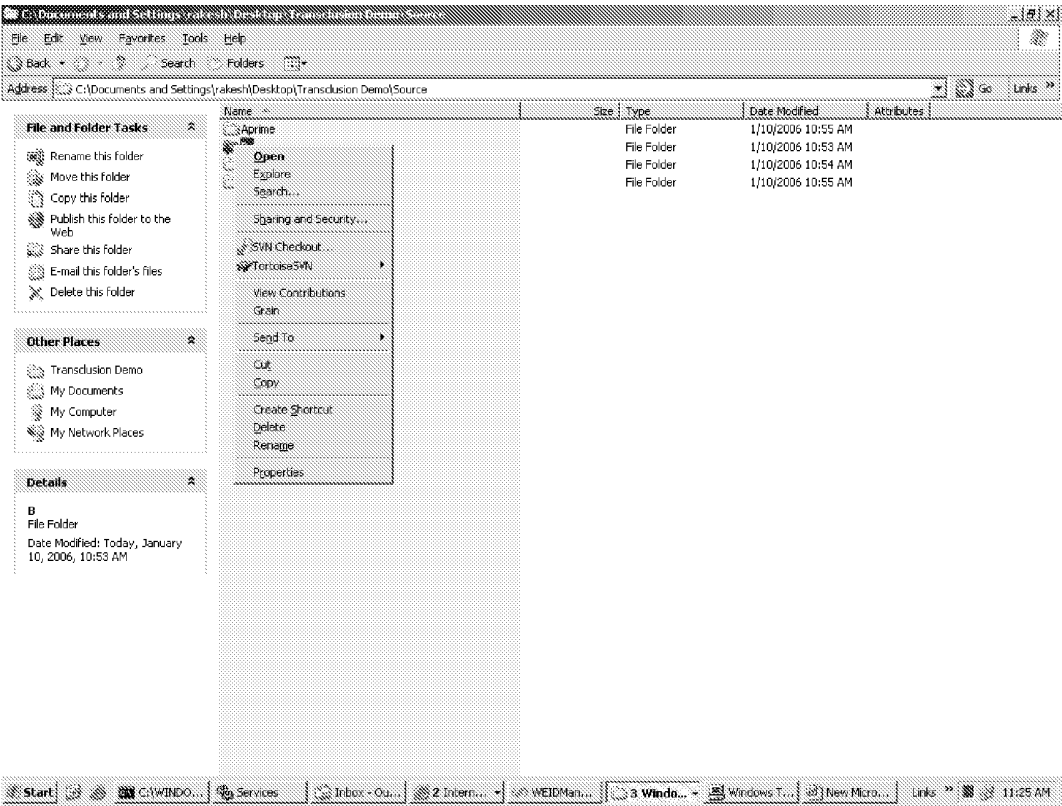


Figure 78

Contributions: After modification of result, view at source side
Selecting the view contributions item

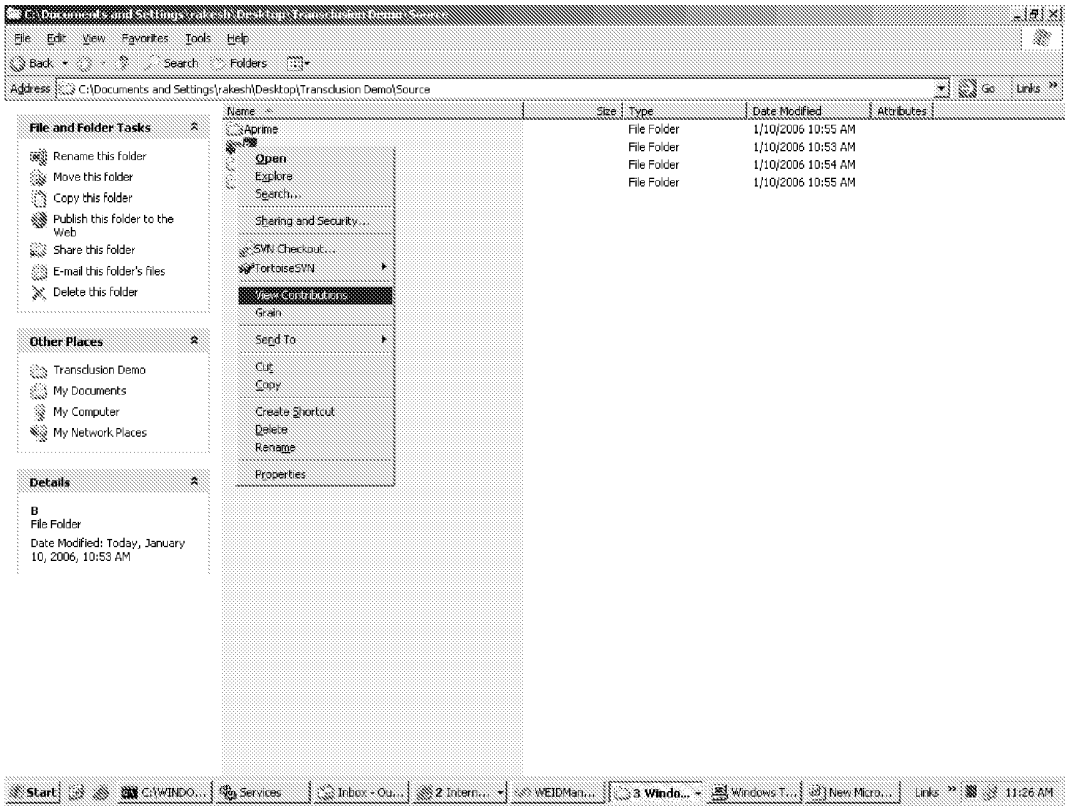


Figure 79

Contributions: After modification of result, view at source side
After selection of the ‘View Contributions’ the contributions are being displayed

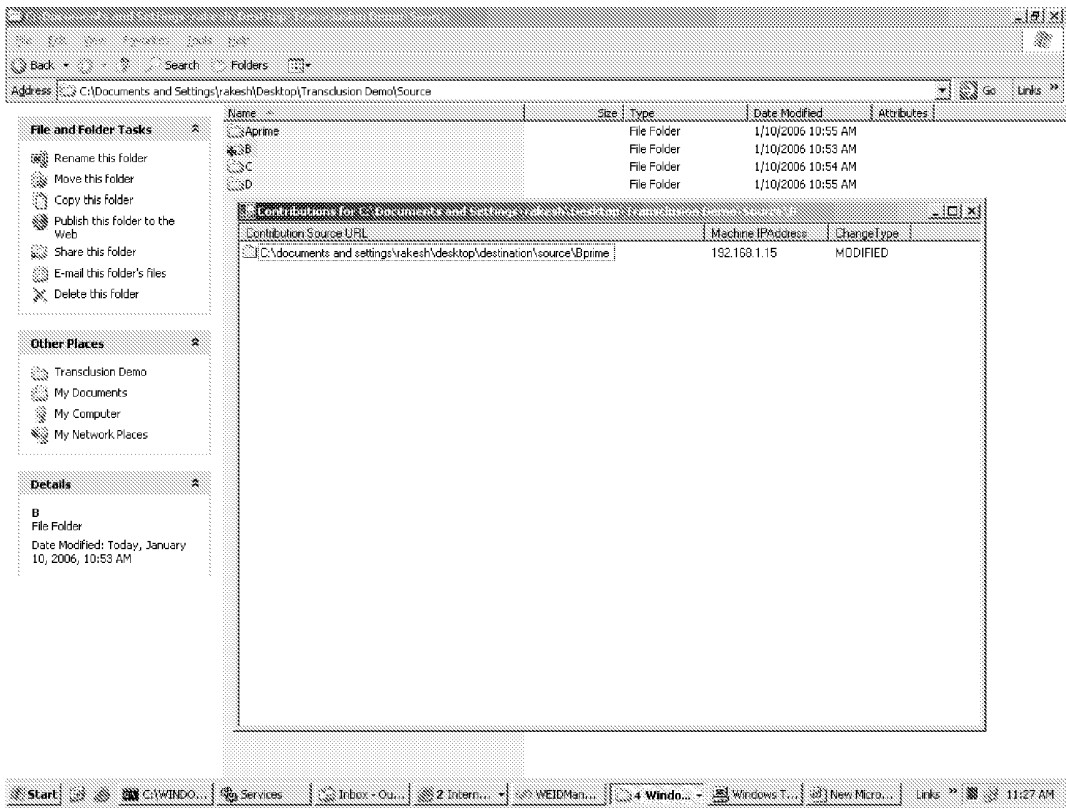


Figure 80

5 **Contributions: After modification of result, view at source side**
 The ‘Accept Contribution’ and ‘Reject Contribution’ contextual menu items are being
 Displayed on right click

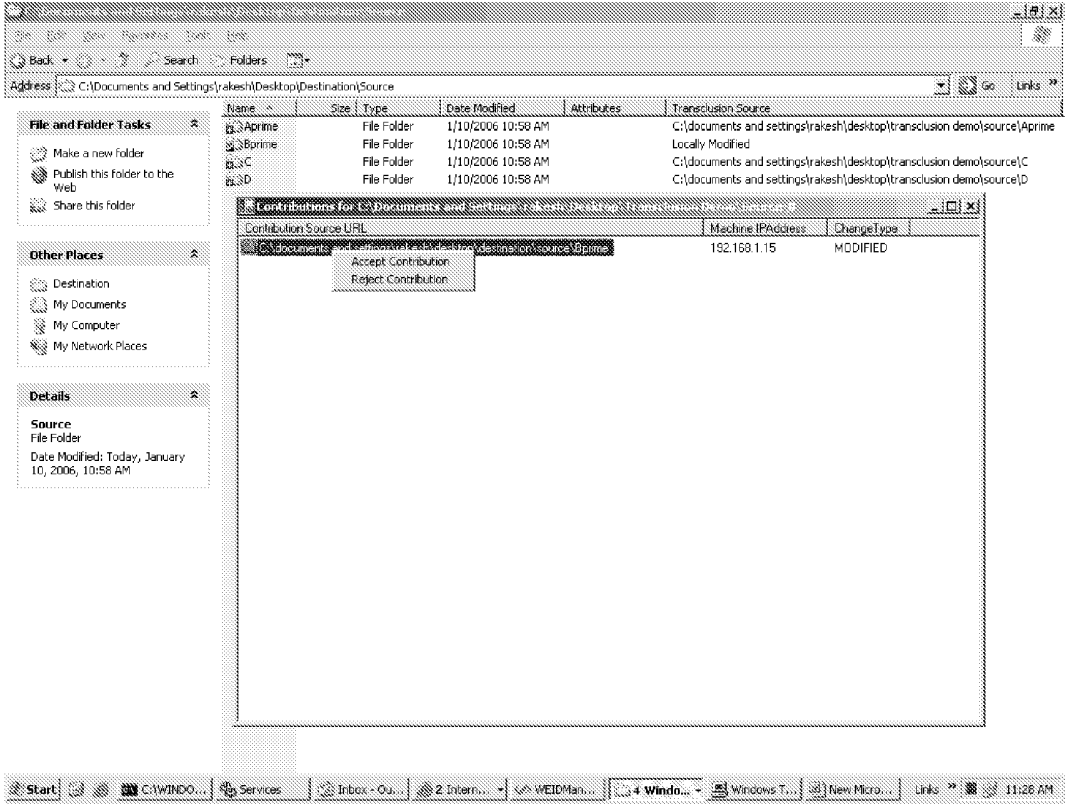


Figure 81

Contributions: After modification of result, view at source side
Selecting the ‘Accept Contributions’ item

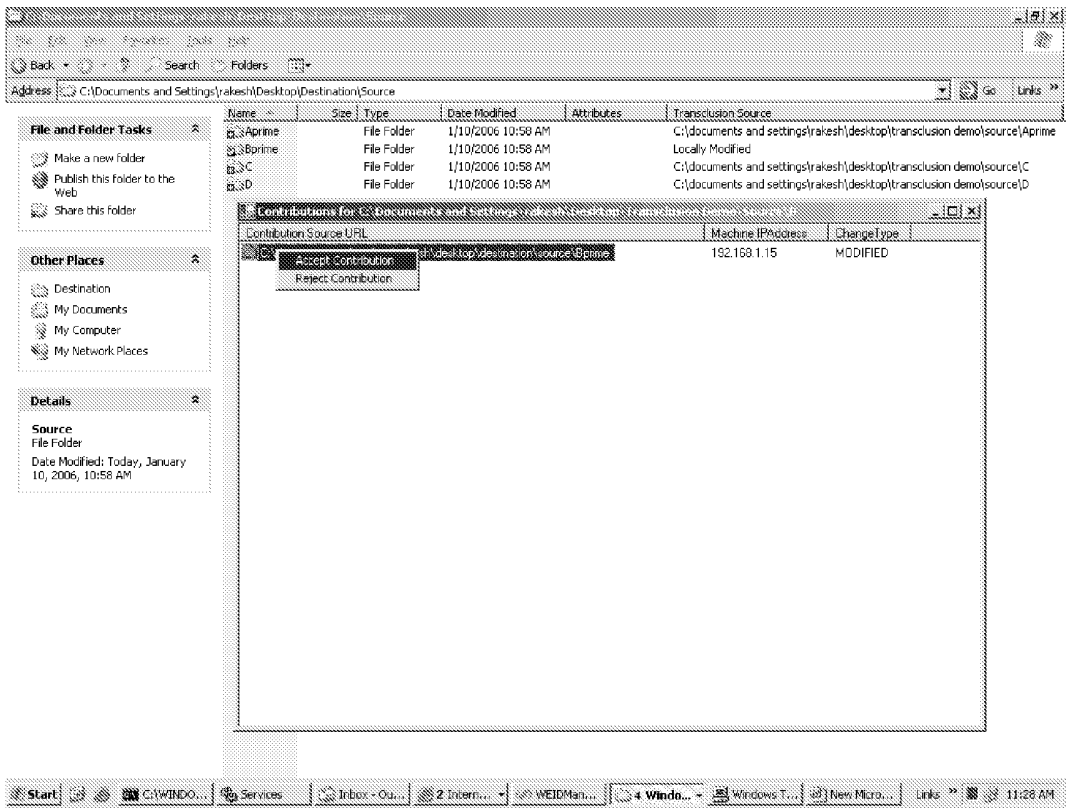


Figure 82

Contributions: After modification of result, view at source side
After selection of Accept Contributions 'B' has changed to 'Bprime'

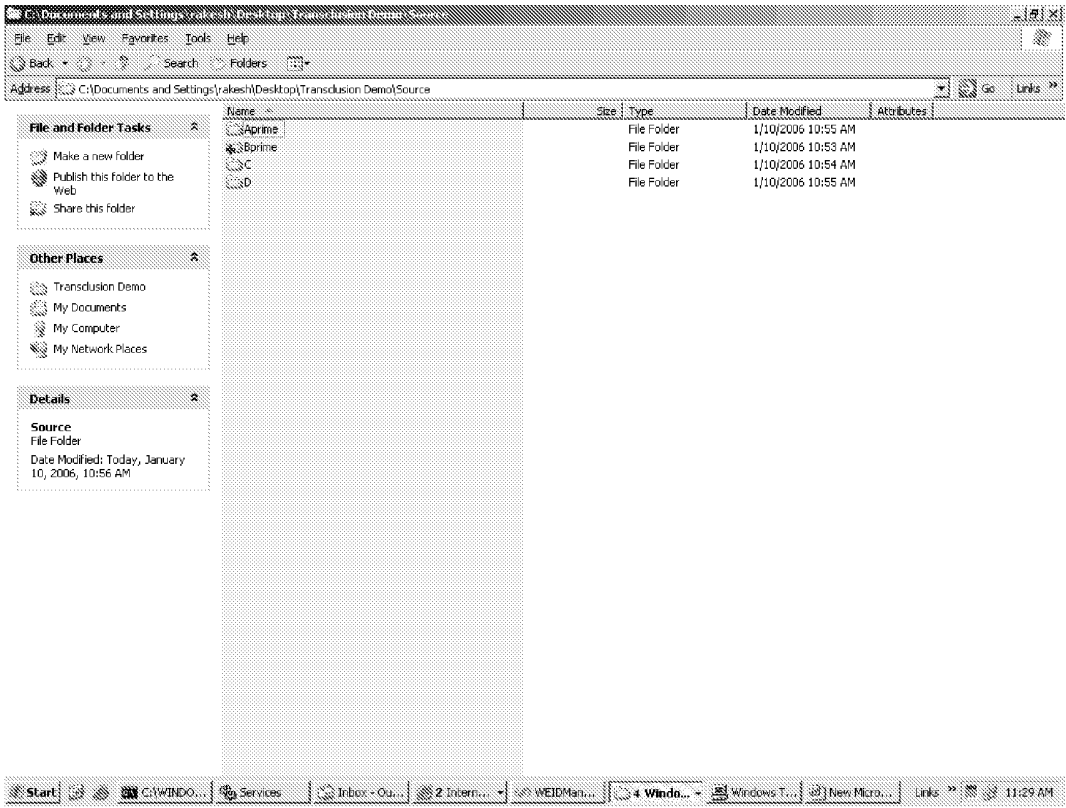
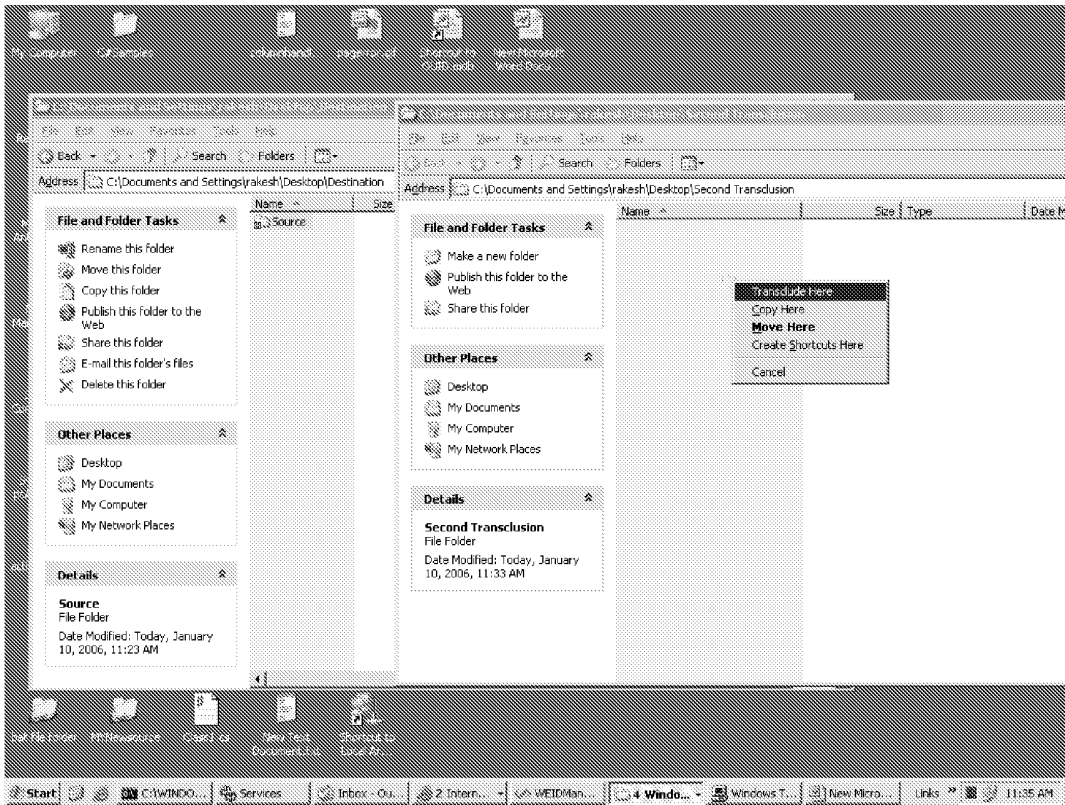


Figure 83

Doing Second Transclusion: Transclusion of the Transclusion result



5

Figure 84

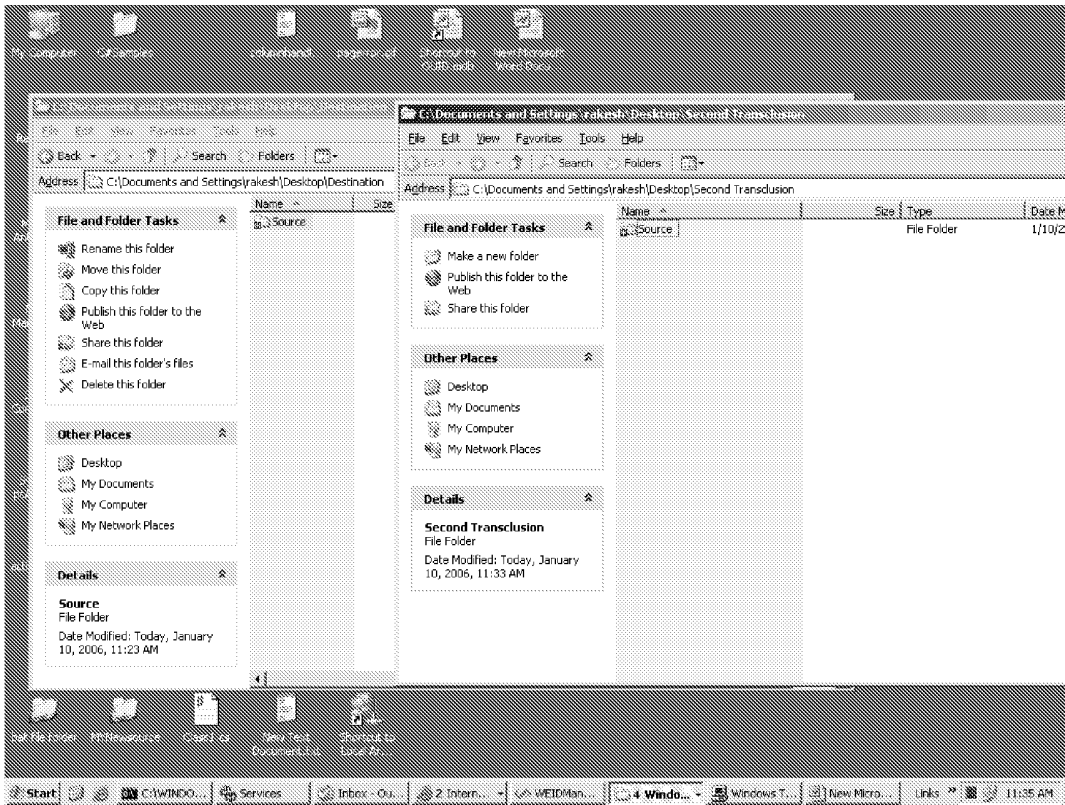
10

15

20

25

Second Transclusion: after the transclusion



5

Figure 85

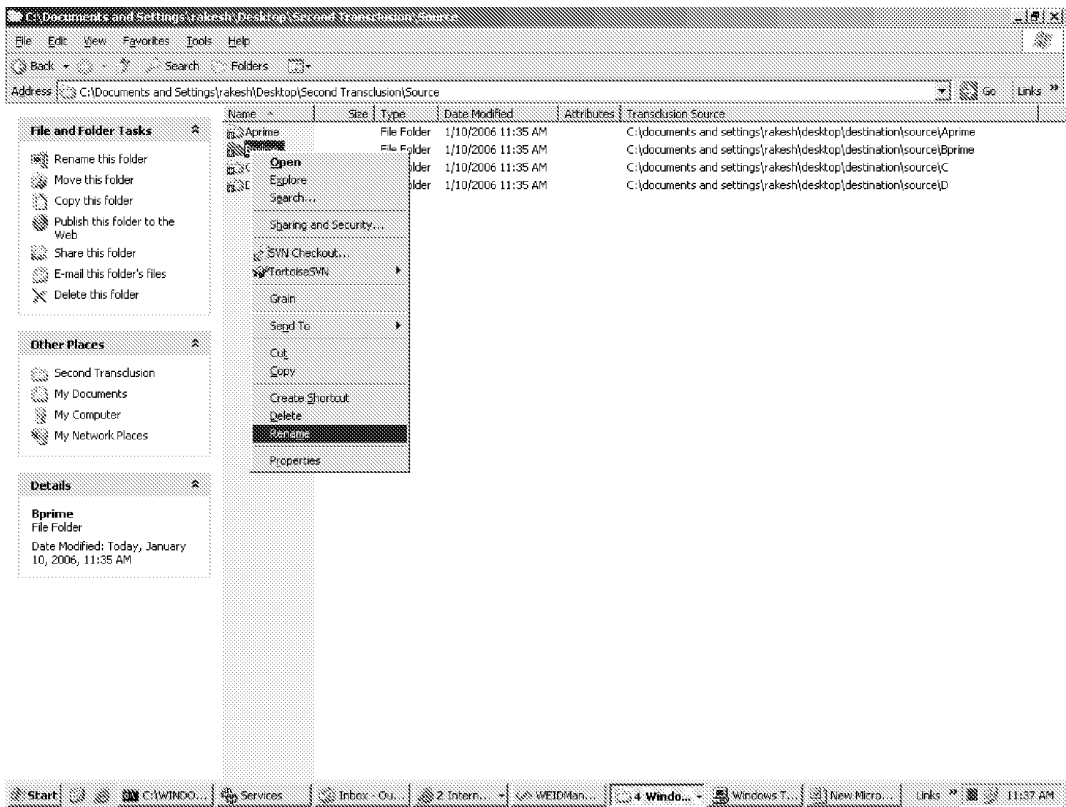
10

15

20

25

Second Transclusion: modifying the result



5

Figure 86

10

15

20

25

Second Transclusion : After modification of result, view at source
The red circle representing the state ‘Transclusion Result modified and having contribution’

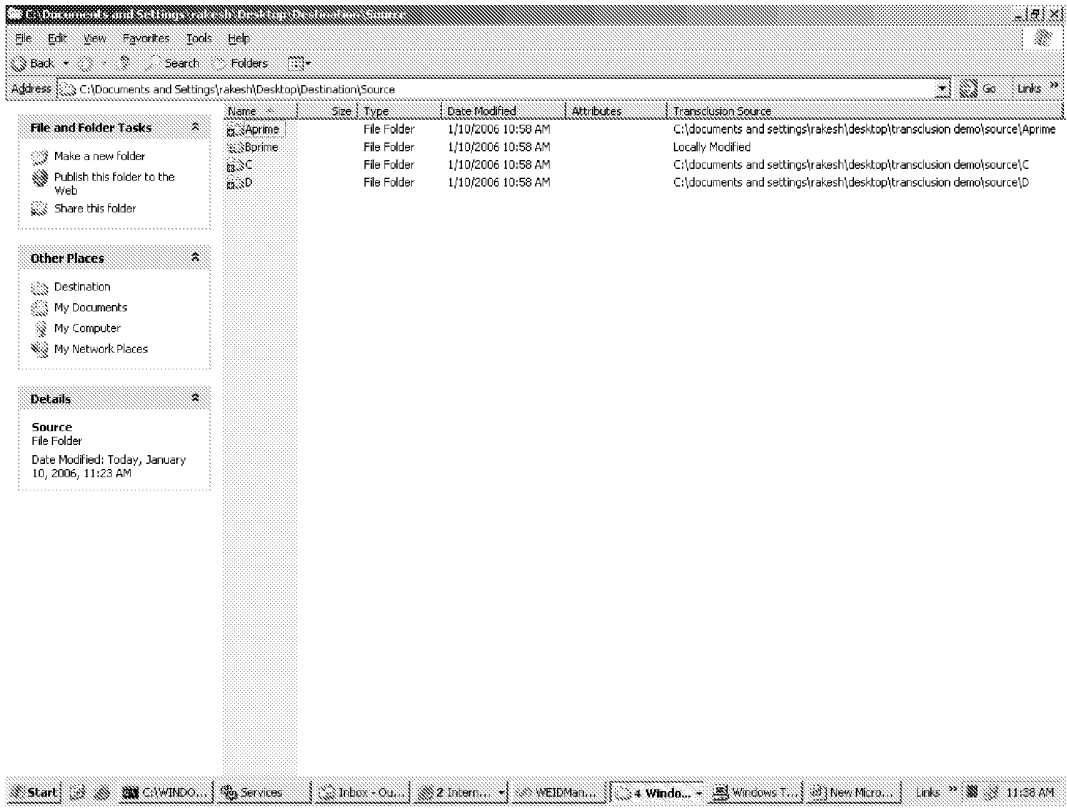
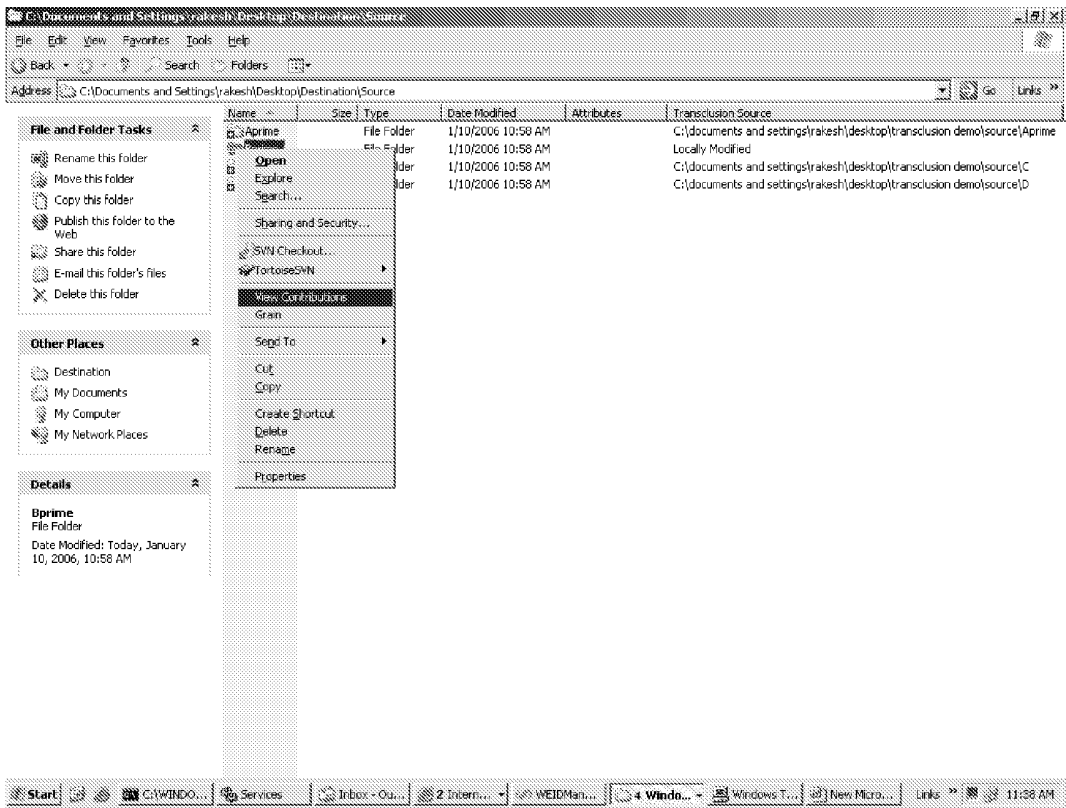


Figure 87

Second Transclusion : After modification of result view at source
The right click contextual menu shows the 'View Contribution' menu item



5

Figure 88

10

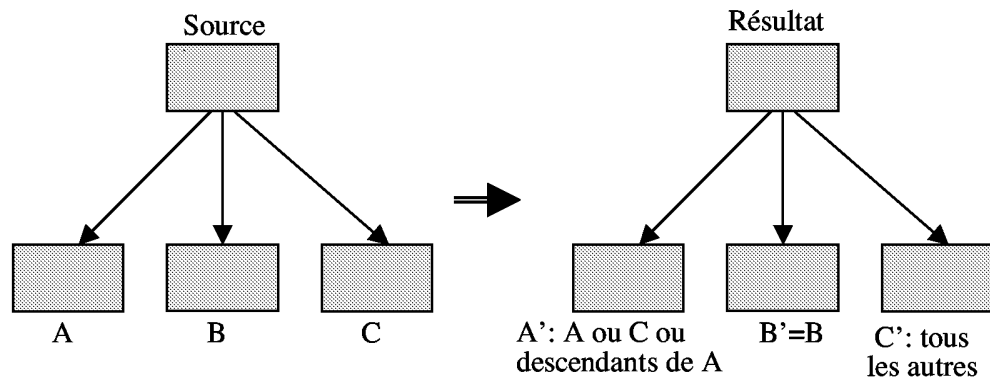


Fig. 89

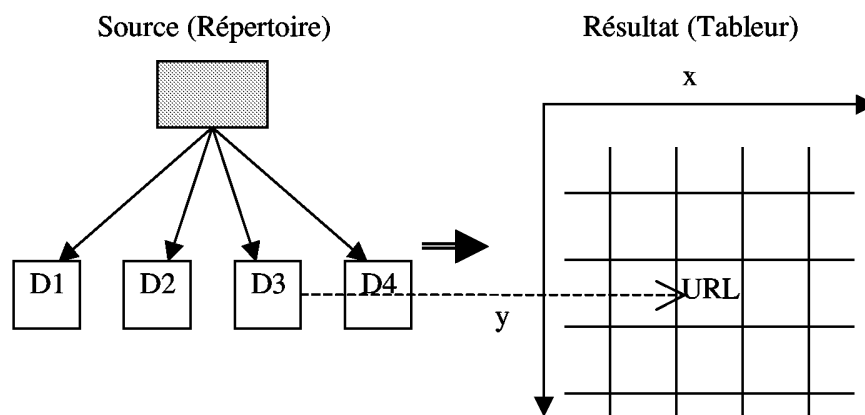


Fig. 90

5

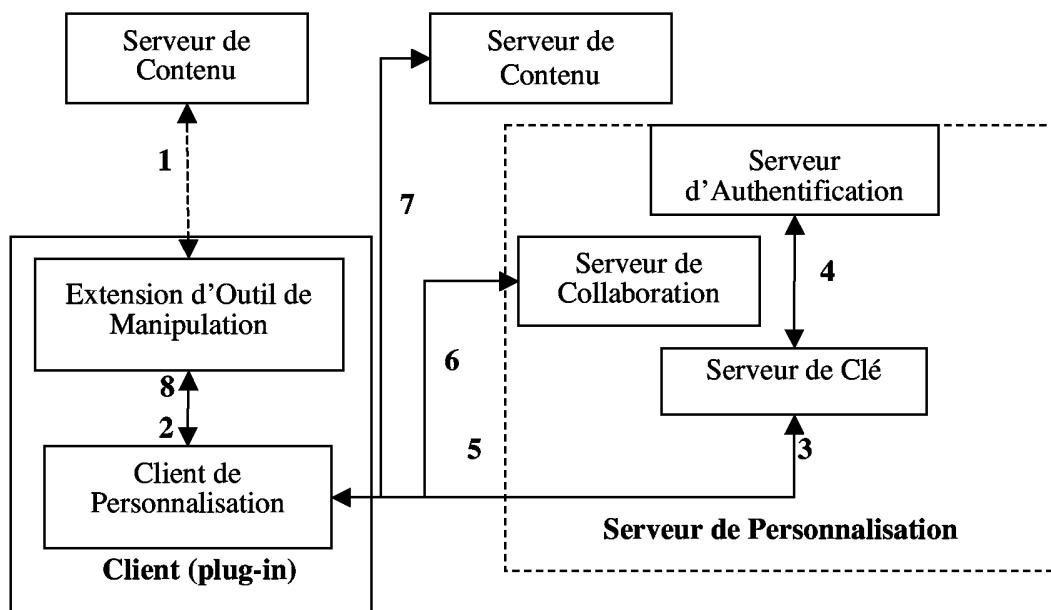


Fig. 91

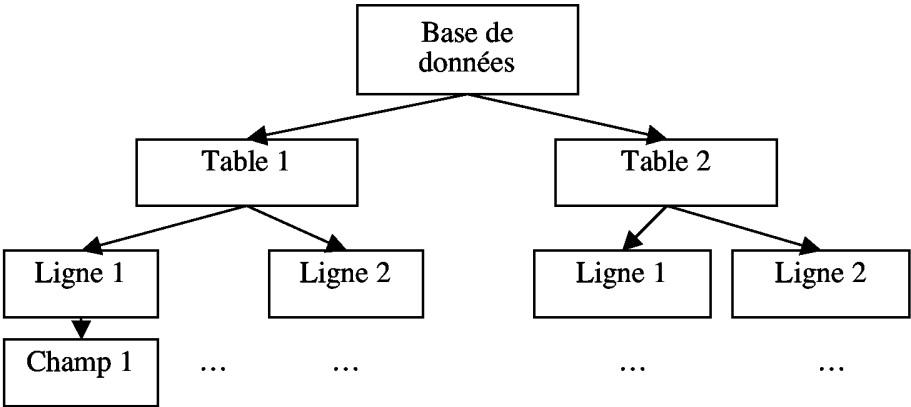


Fig. 92

<http://www.site.org/page/>

	E1	☐	Recevoir	

Fig. 93

5

<http://www.site.org/page/>

	E1	☐	Modifier	

Fig. 94

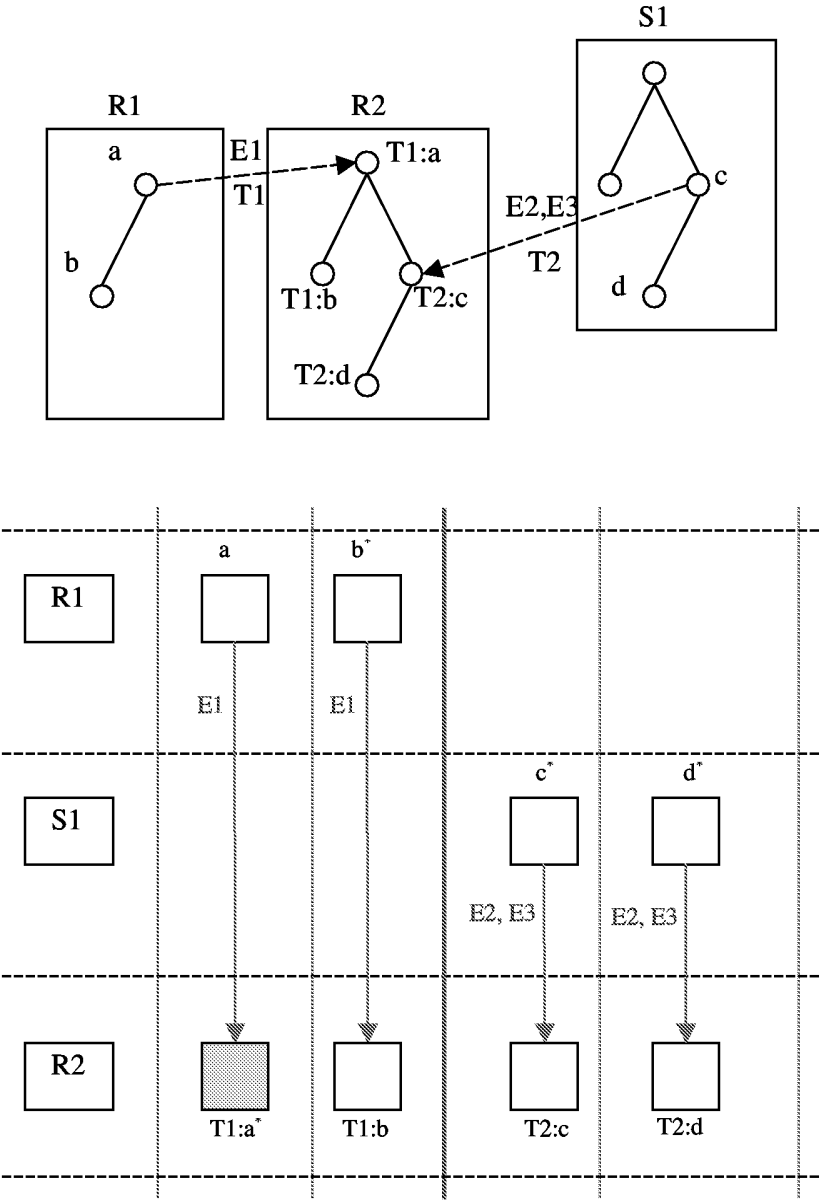
http://www.site.org/page/			
E1	☐	Enregistrer	

Fig. 95

	Accept	☐	Reject
	Accept all		

☐ ☐ ☒ ☐

Fig. 96



5

Figure 97 : R2 dérive de R1 dans E1 et transclut de S1 dans E2 et E3

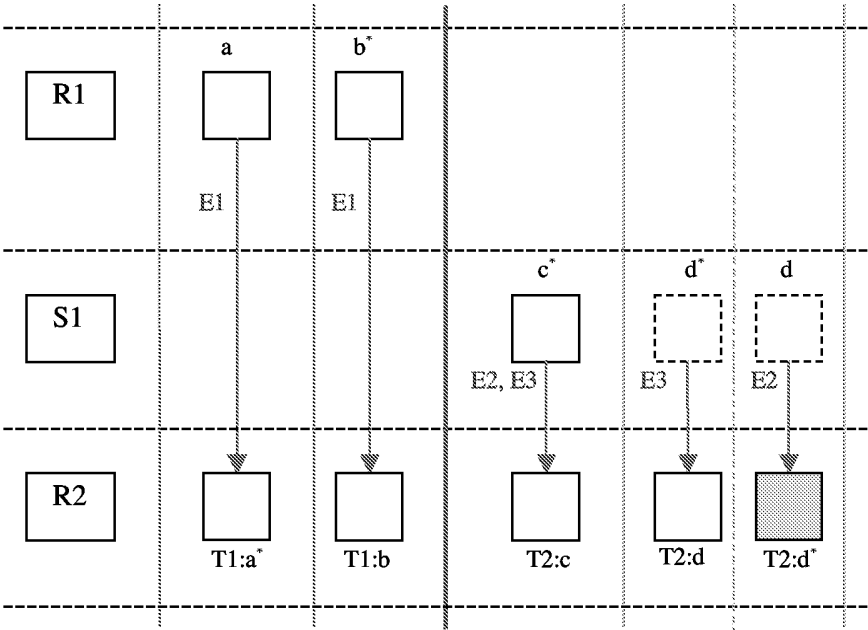


Figure 98: R2 ouvert avec E3 en mode secret ; T2:d est modifié

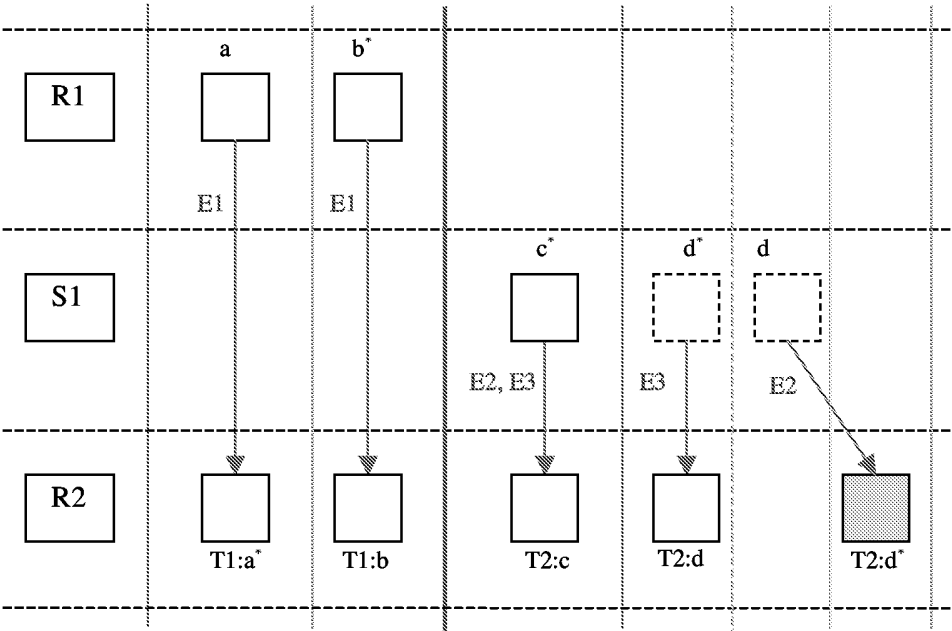


Figure 99: S1 refuse T2:d dans E2

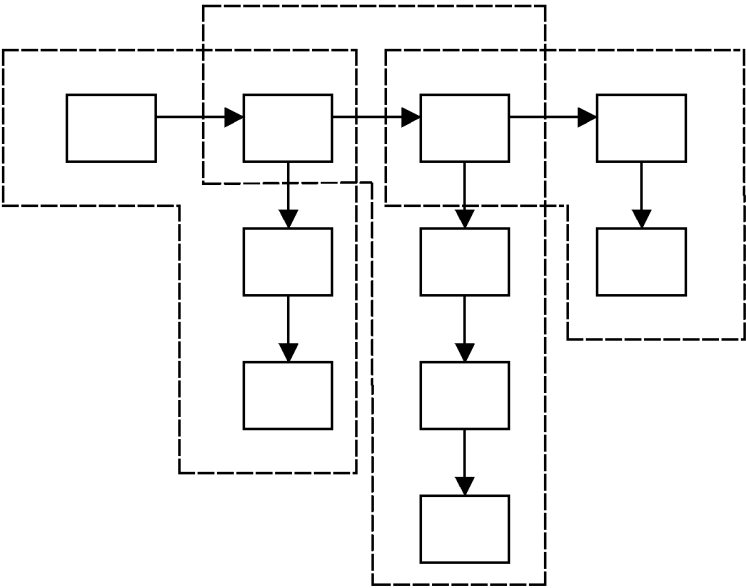


Fig. 100

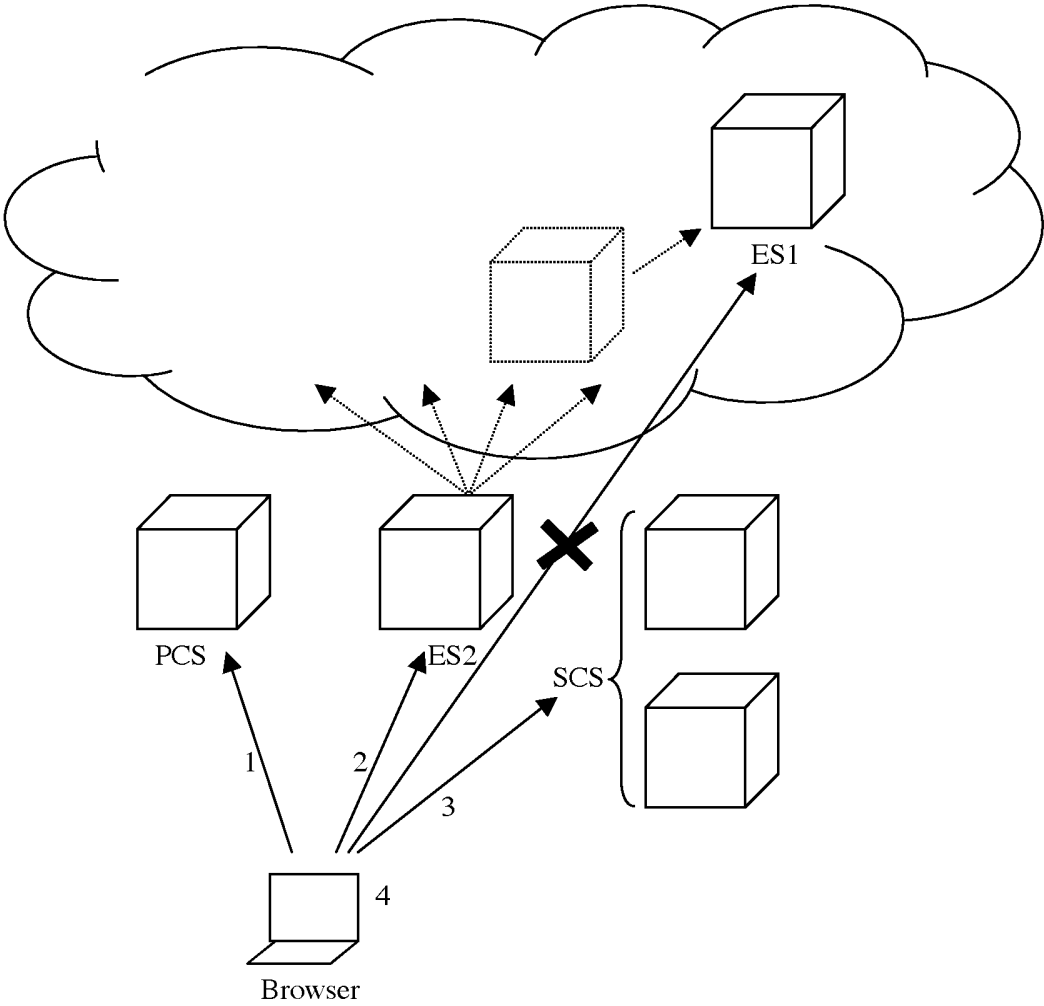


Fig. 101

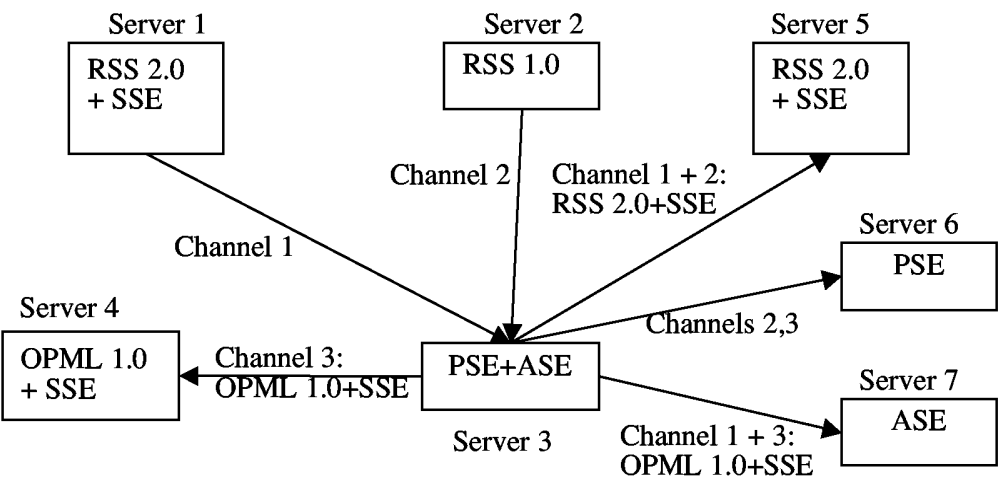


Figure 102

5

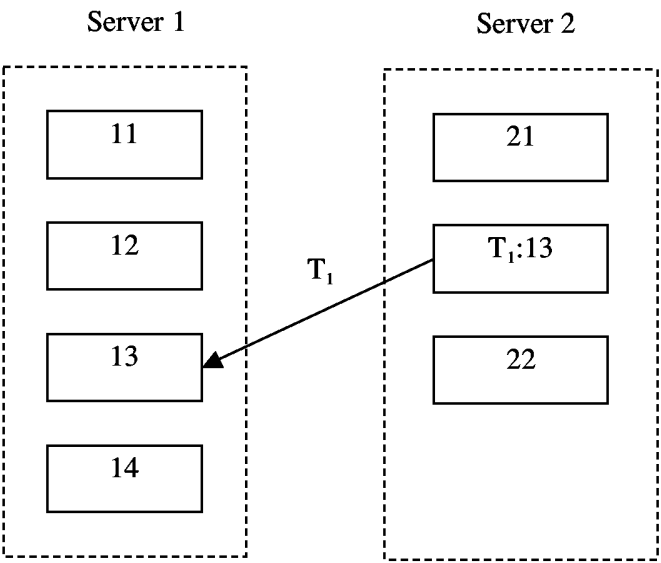


Figure 103

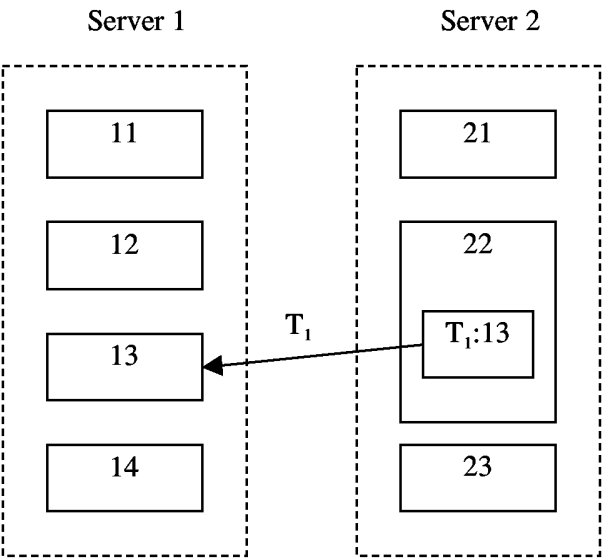


Figure 104

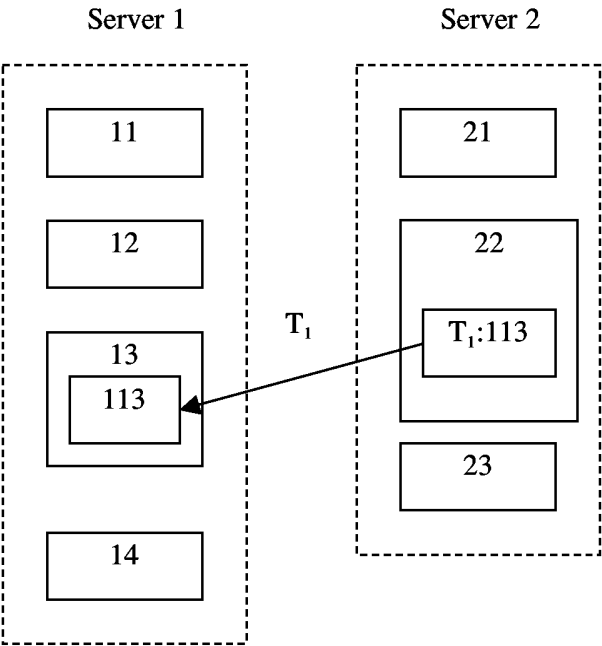


Figure 105

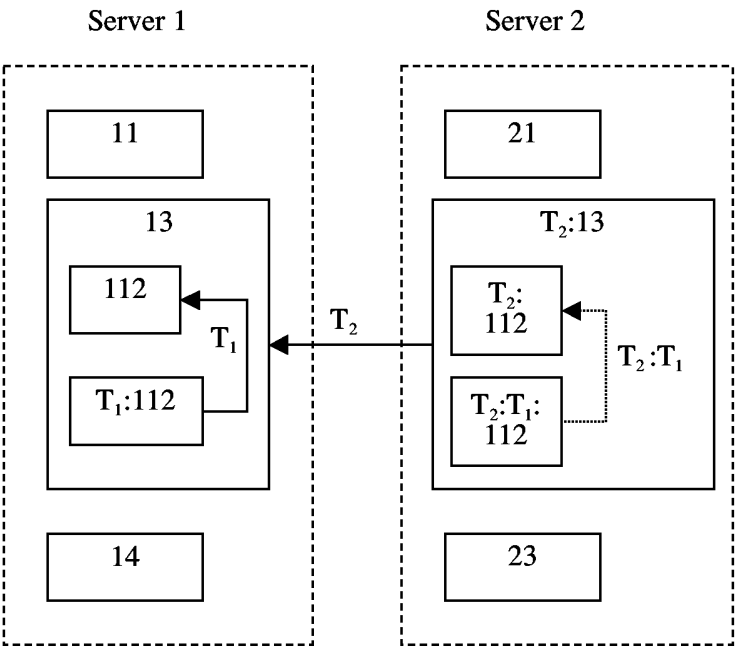


Figure 106

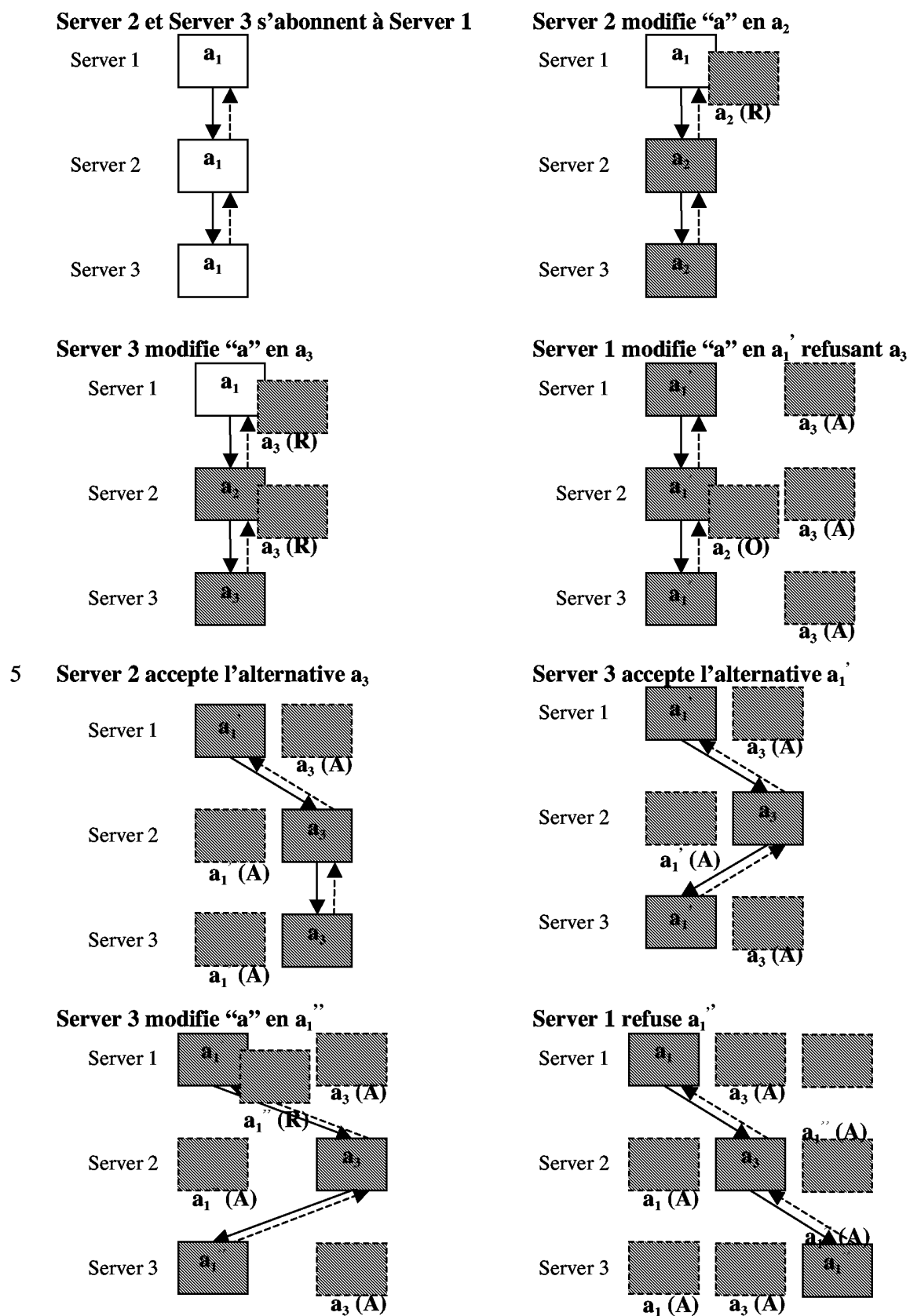


Figure 107

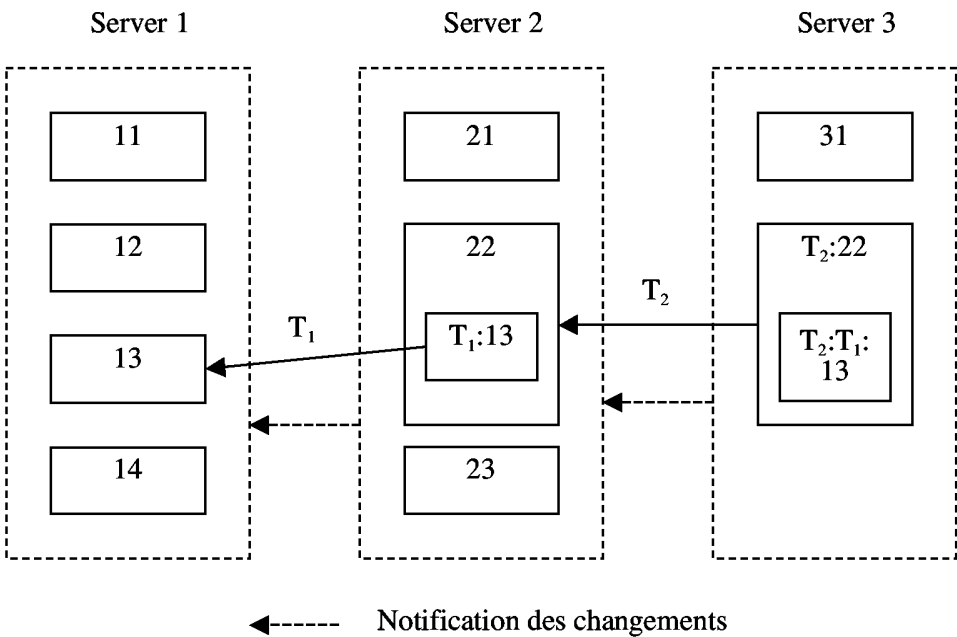
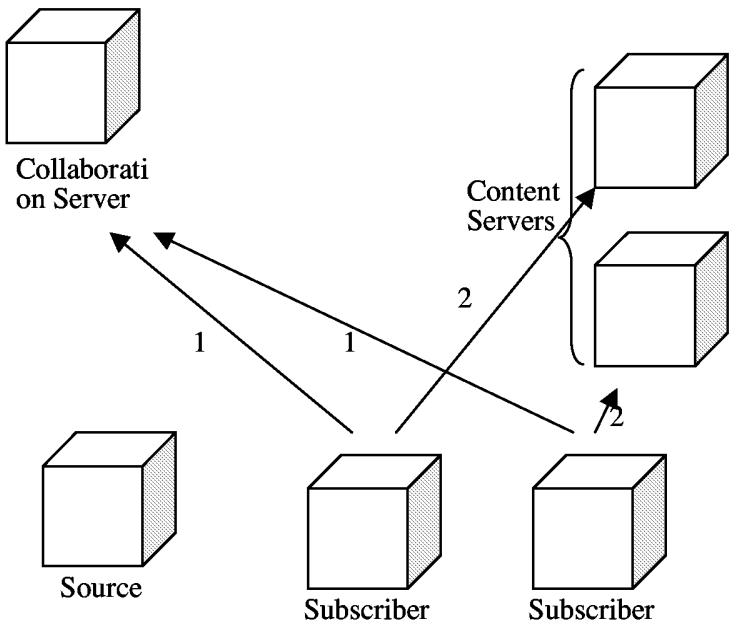


Figure 108



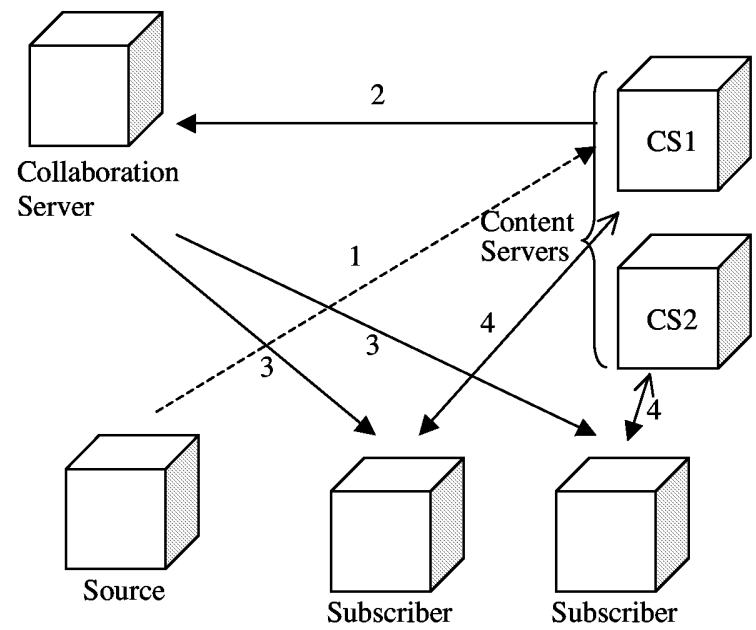


Figure 110

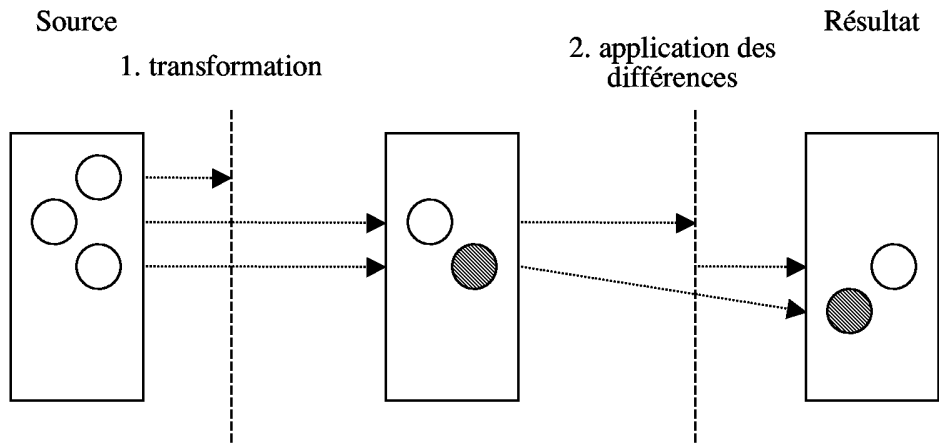


Figure 111

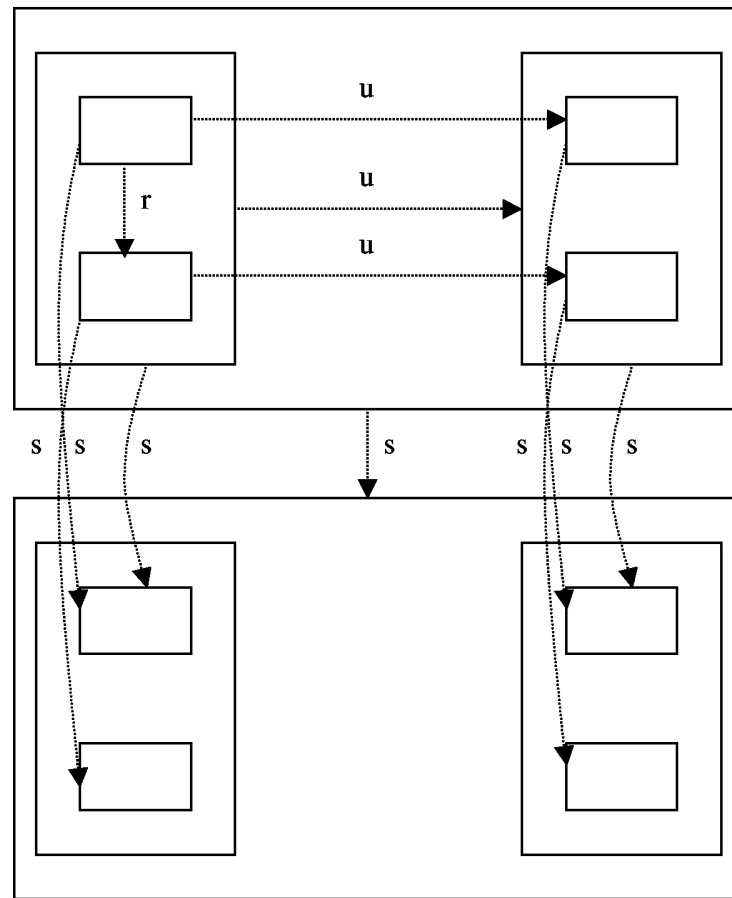


Figure 112

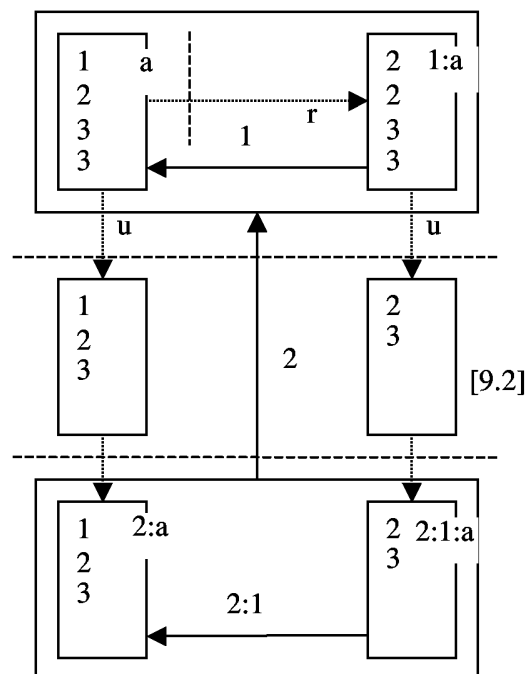


Figure 113

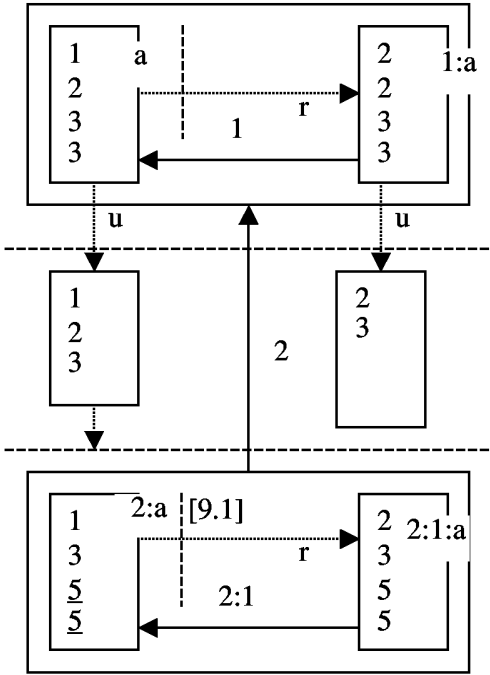


Figure 114

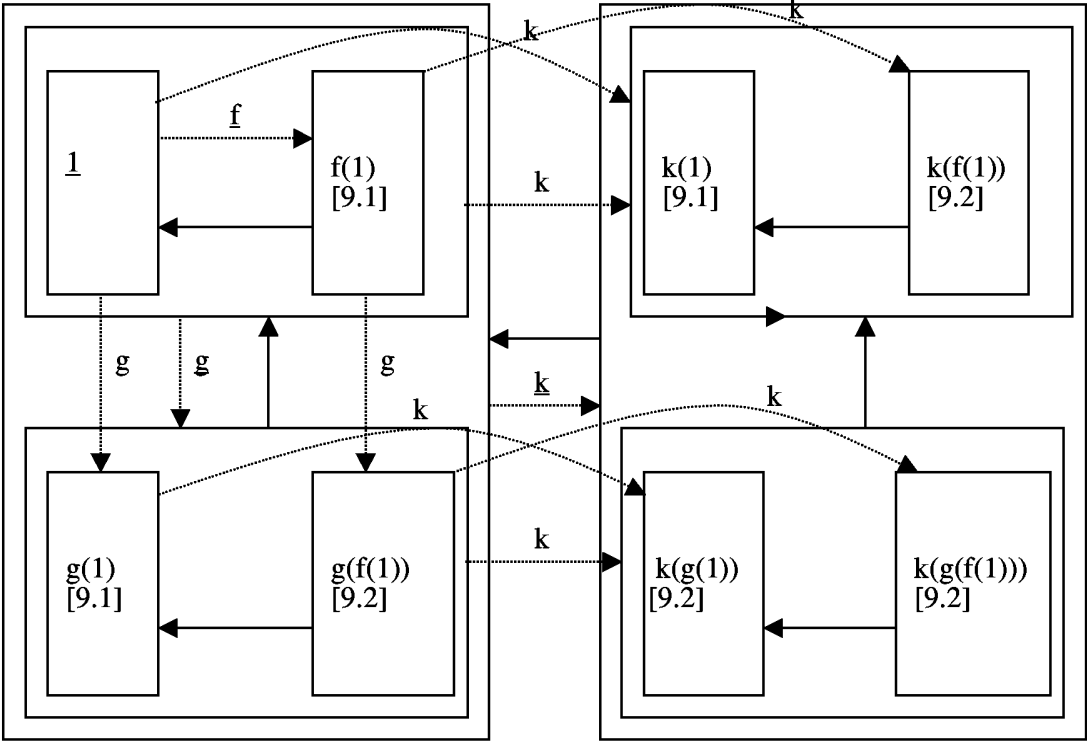


Figure 114

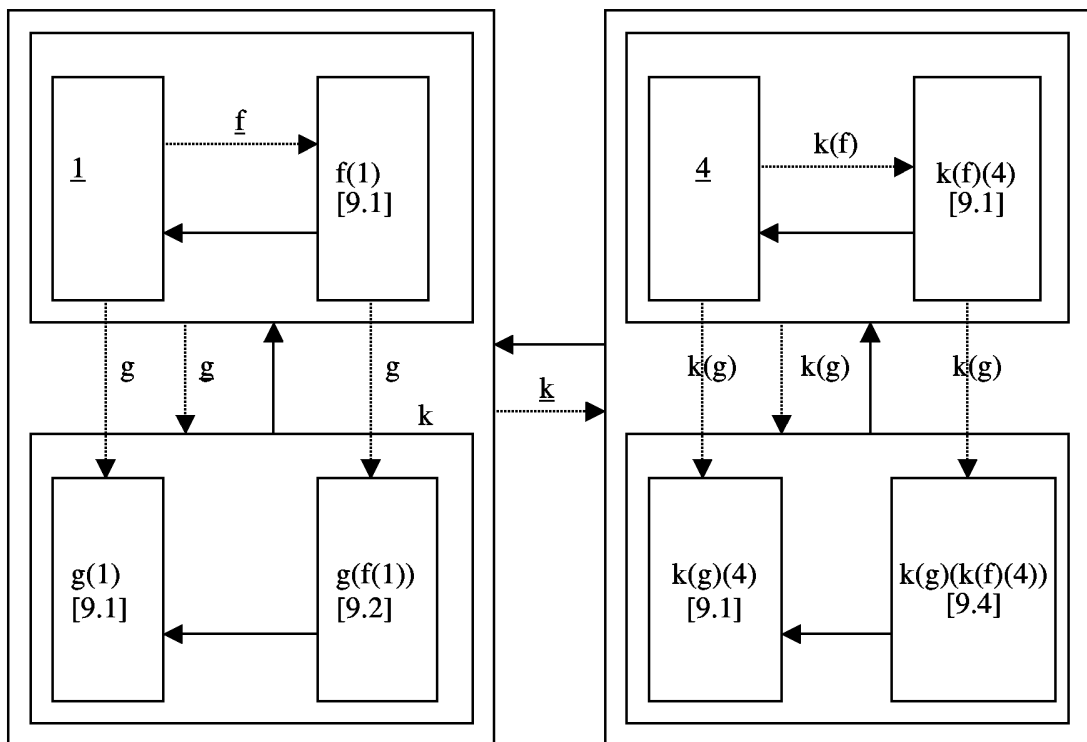


Figure 115

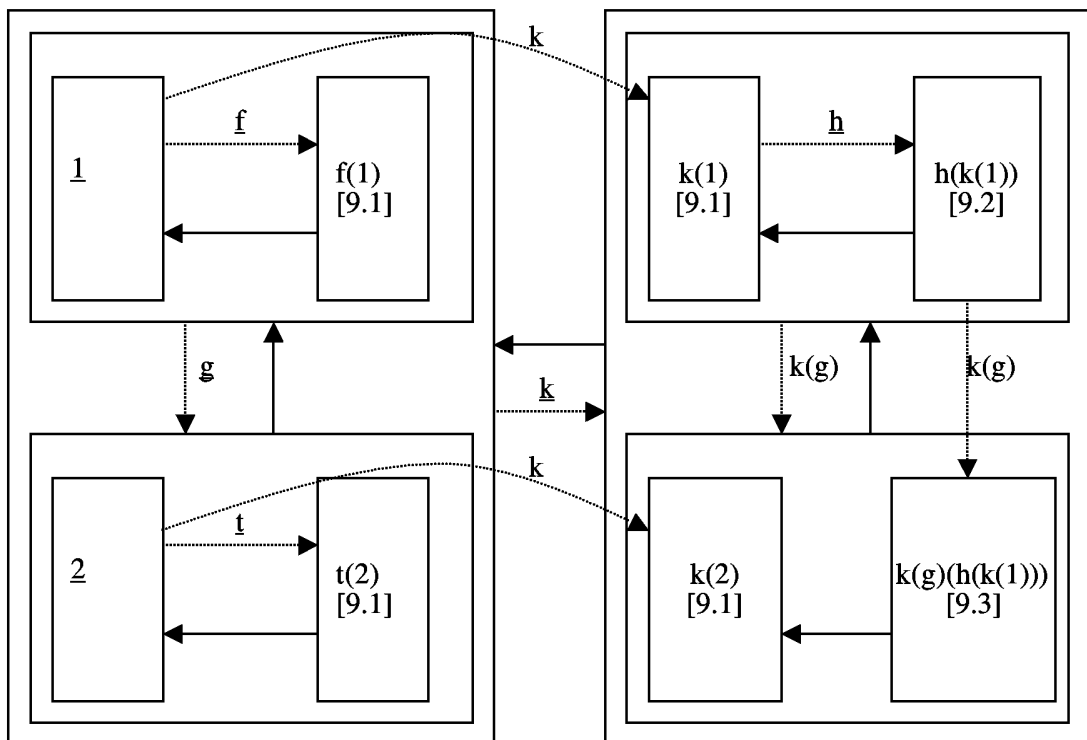


Figure 116

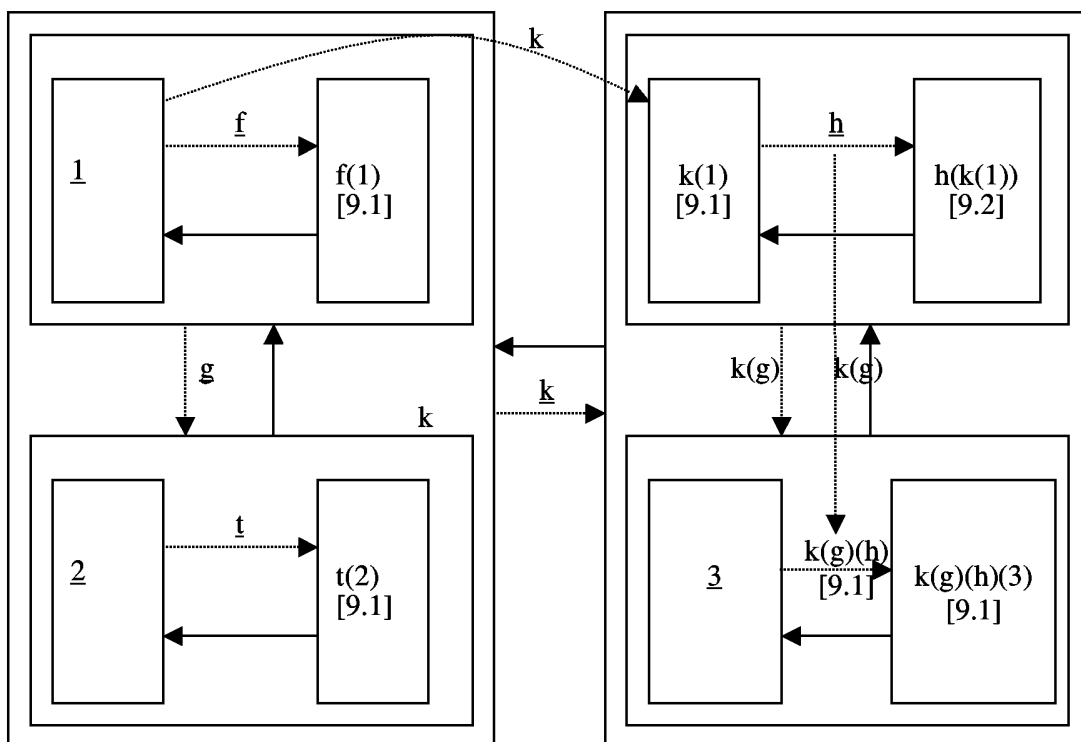


Figure 117

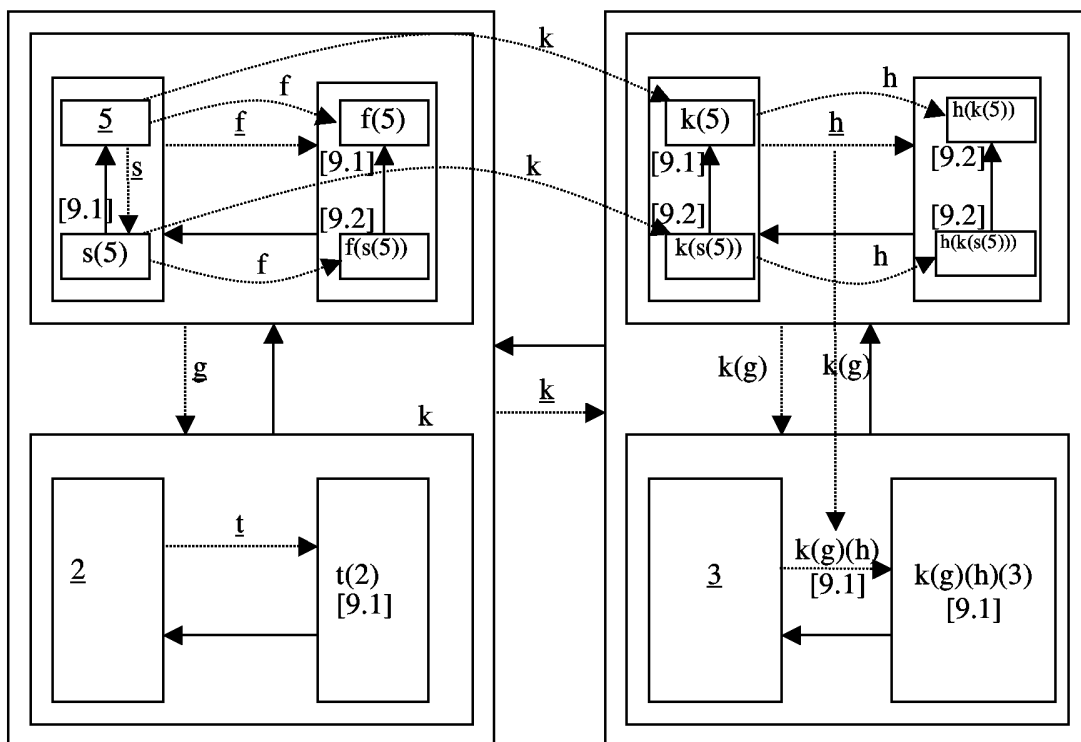


Figure 118

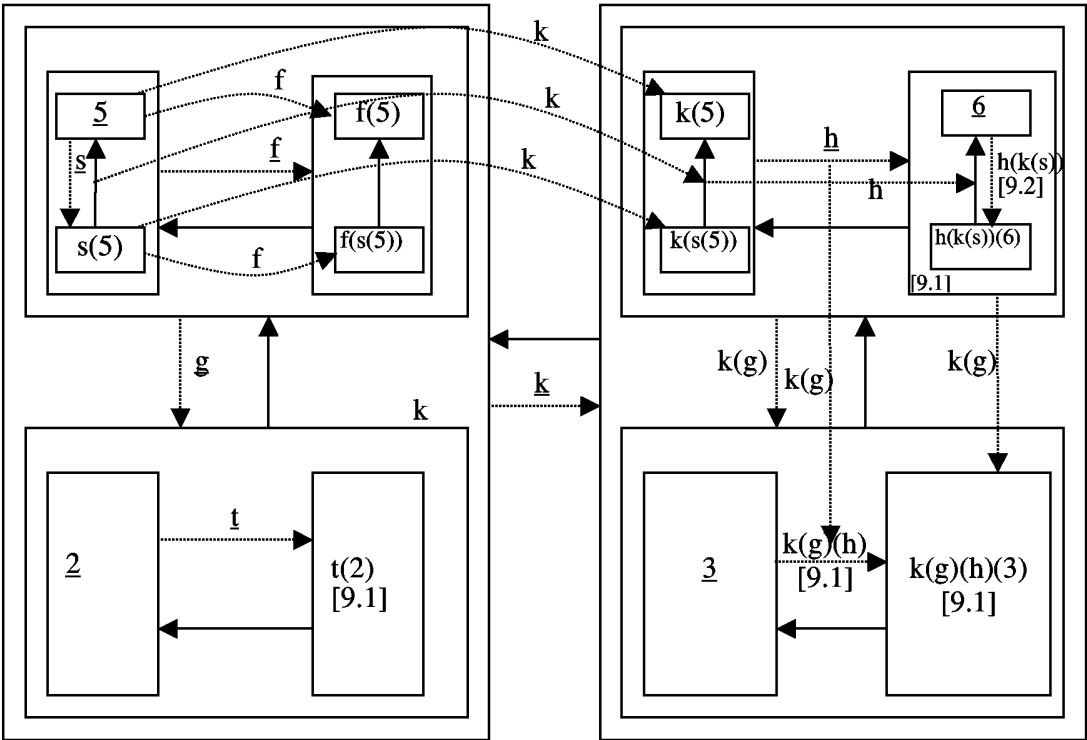


Figure 119

5

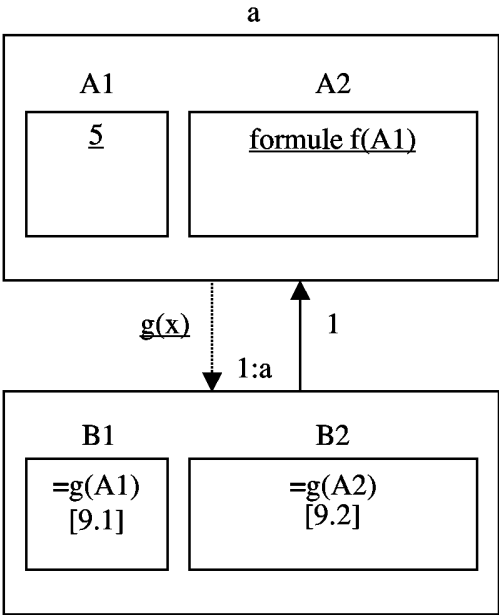


Figure 120

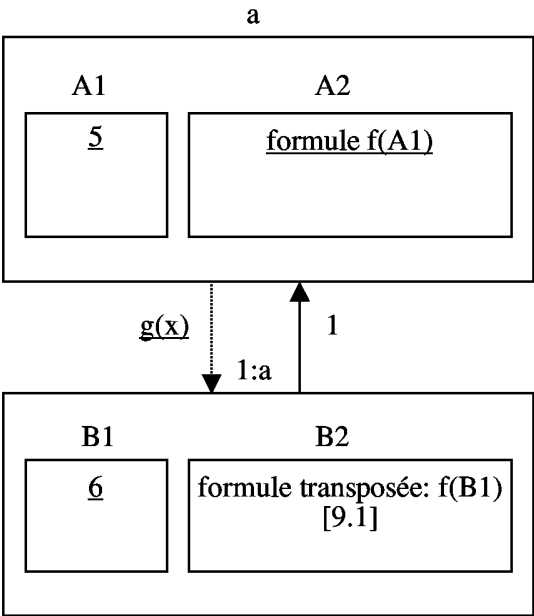


Figure 121

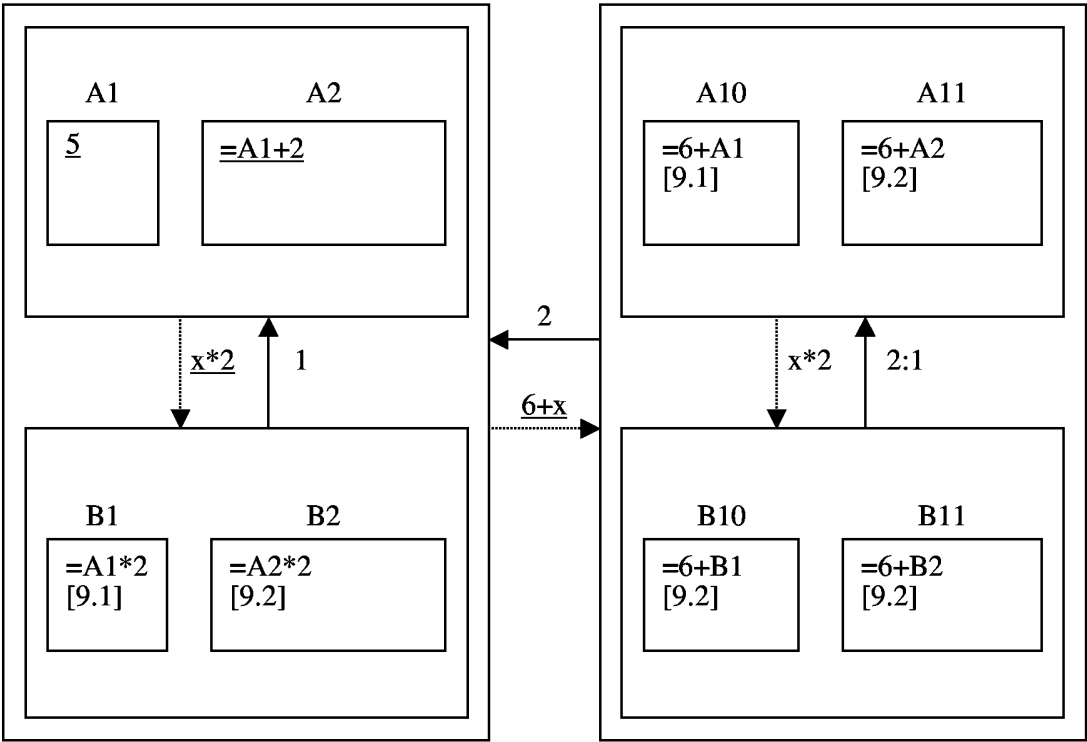


Figure 122

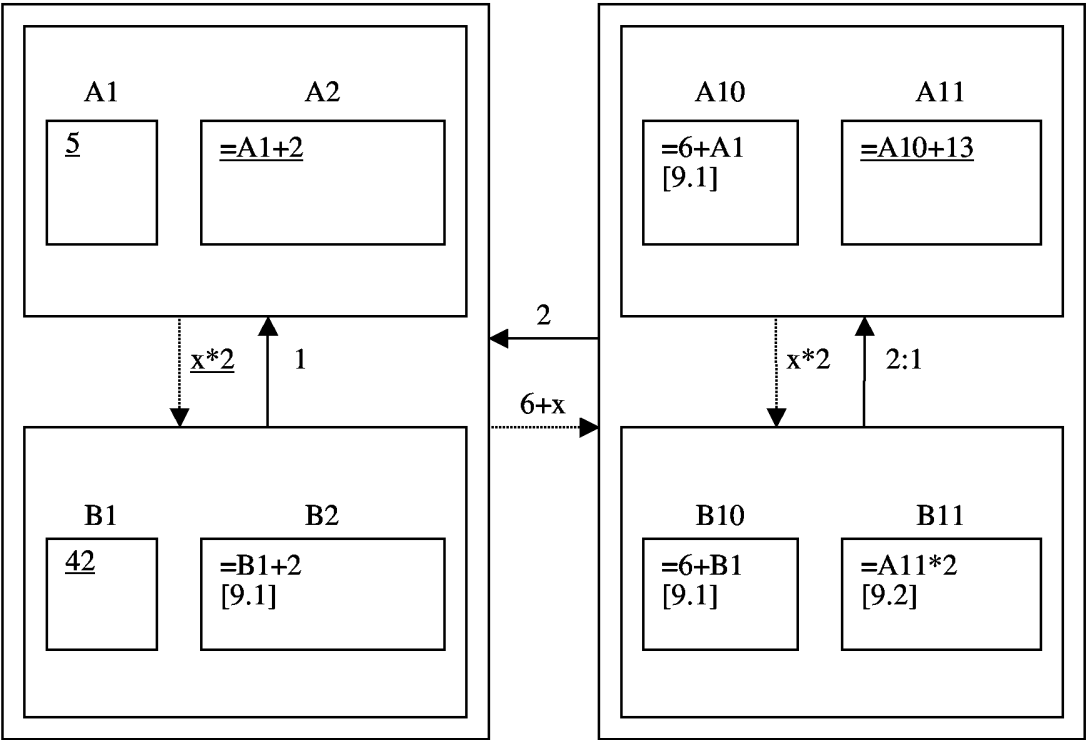


Figure 123

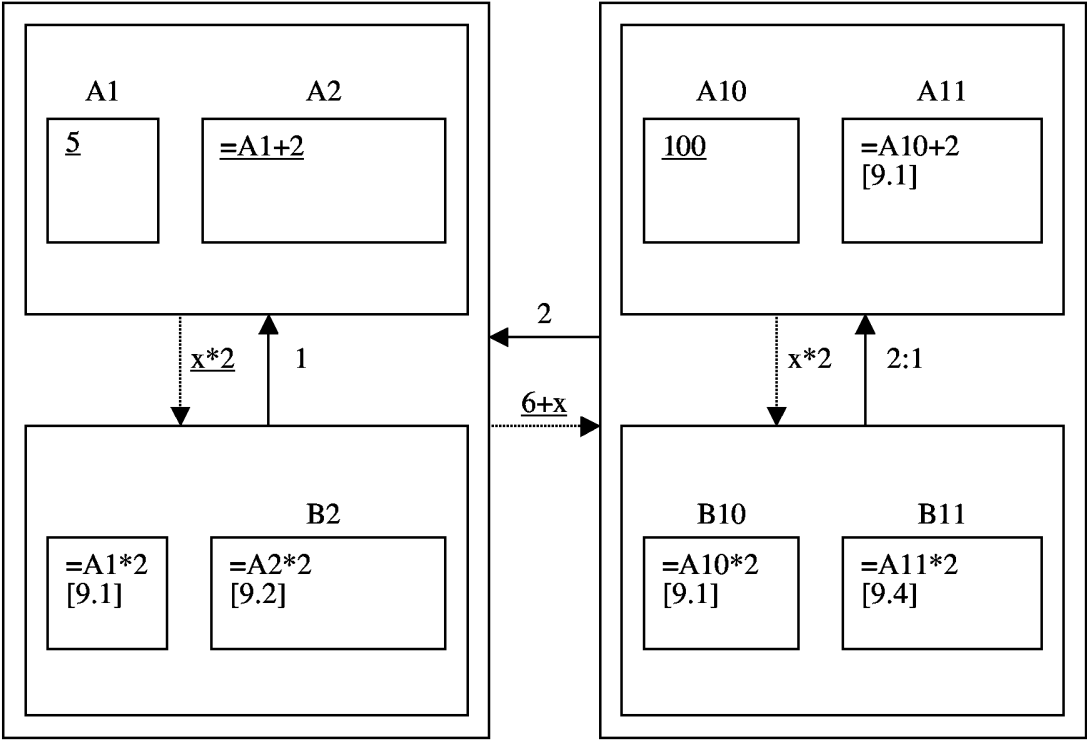


Figure 124

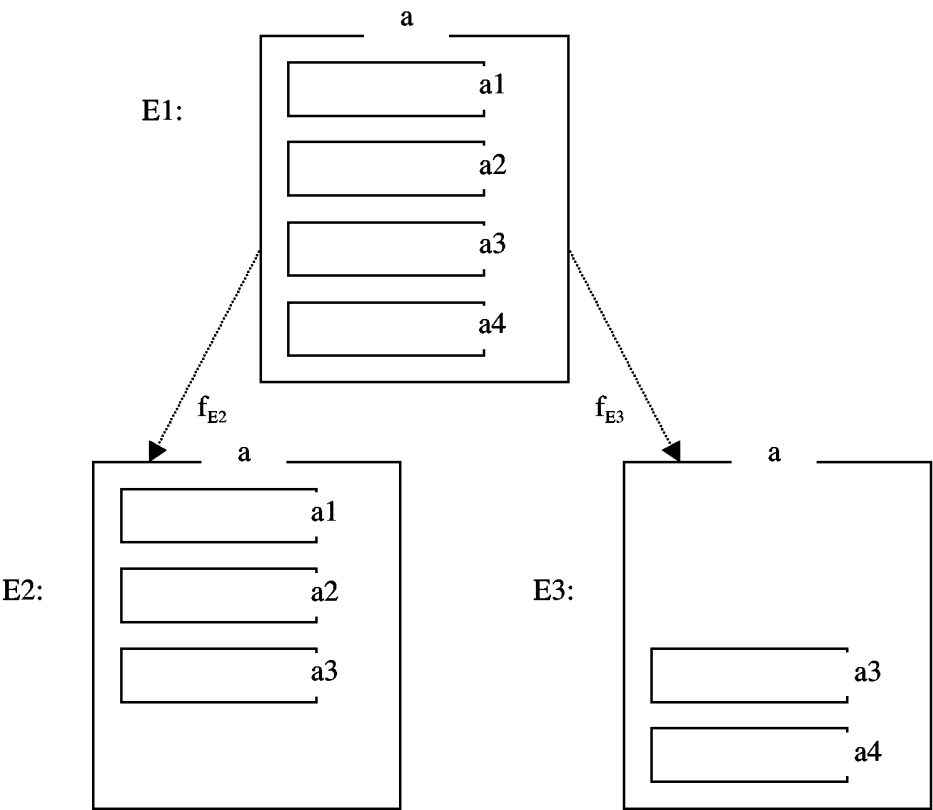


Figure 125

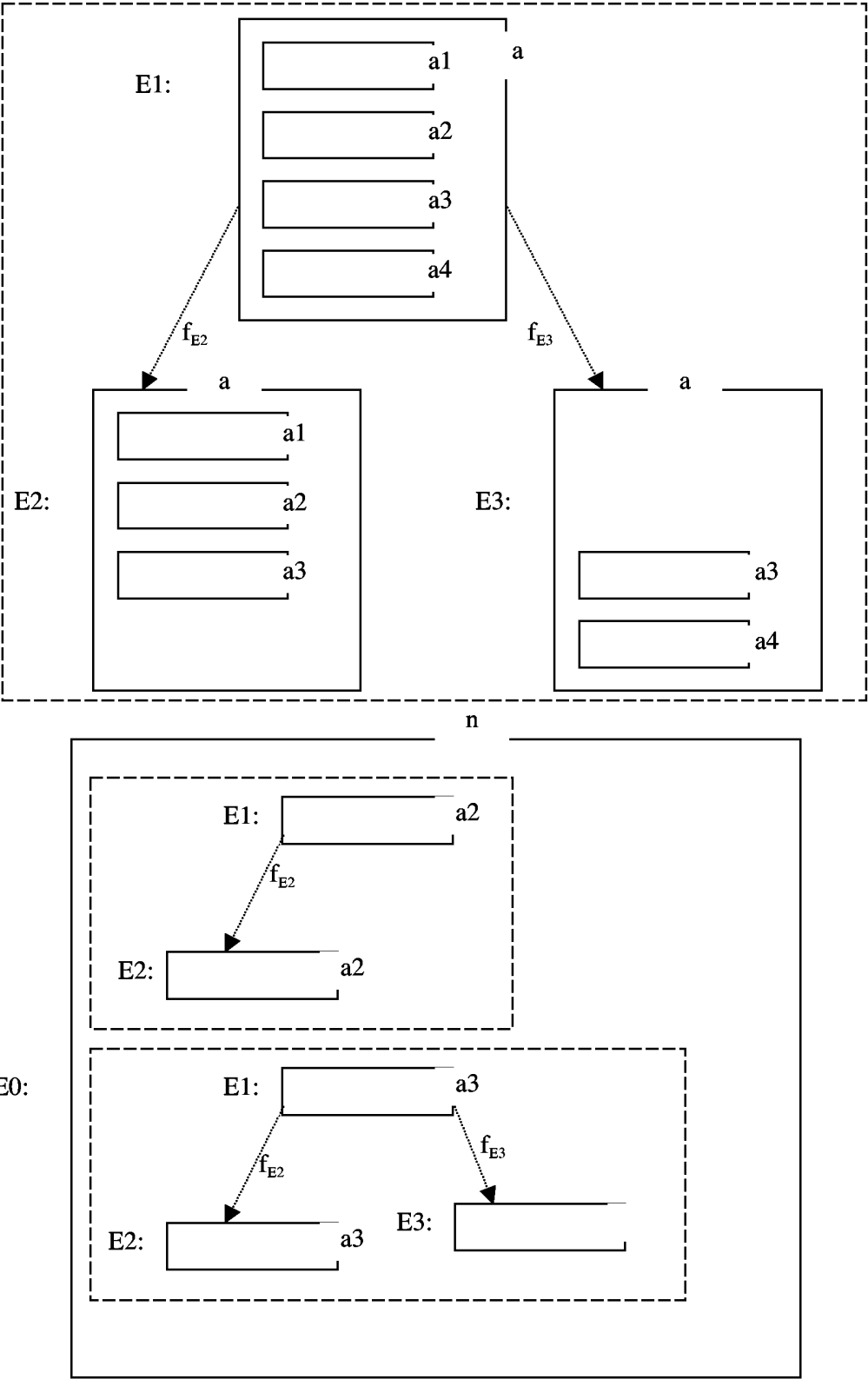


Figure 126

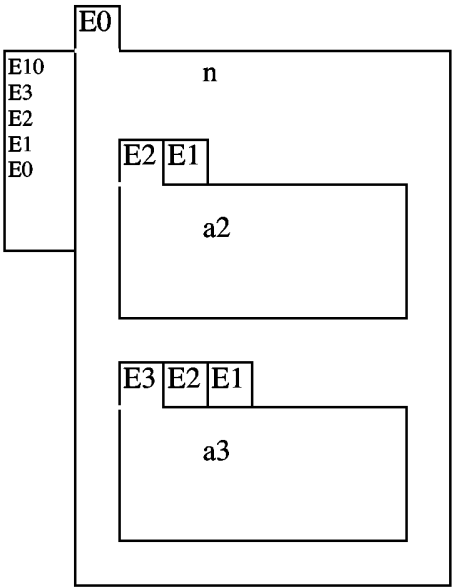


Figure 127

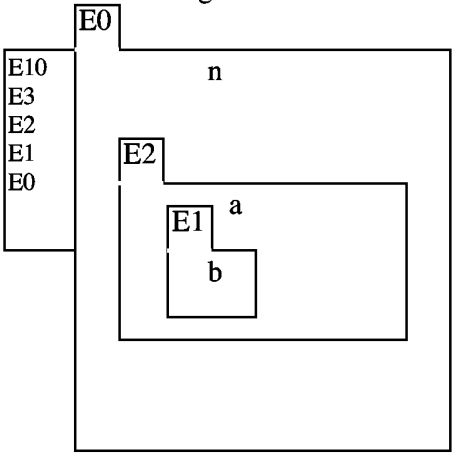


Figure 128

5

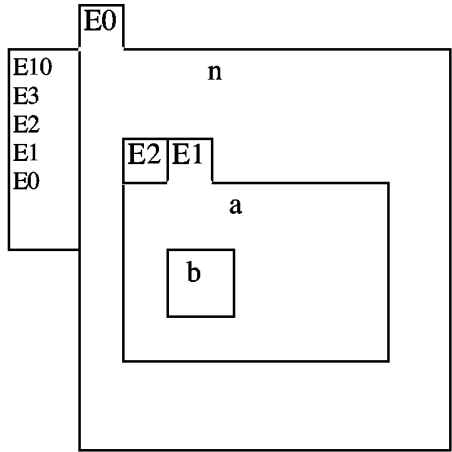


Figure 129

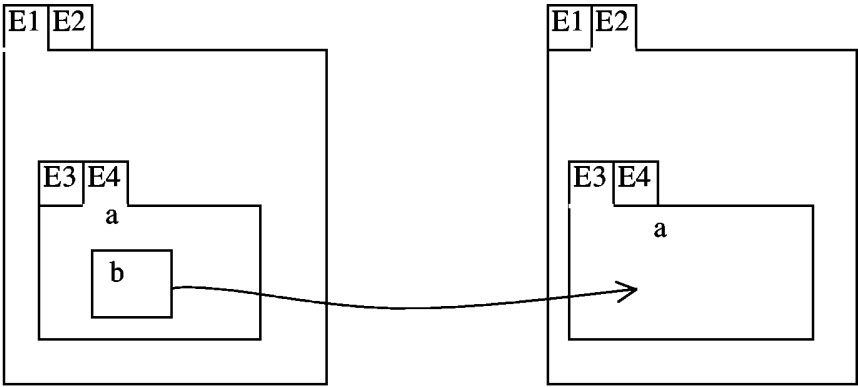
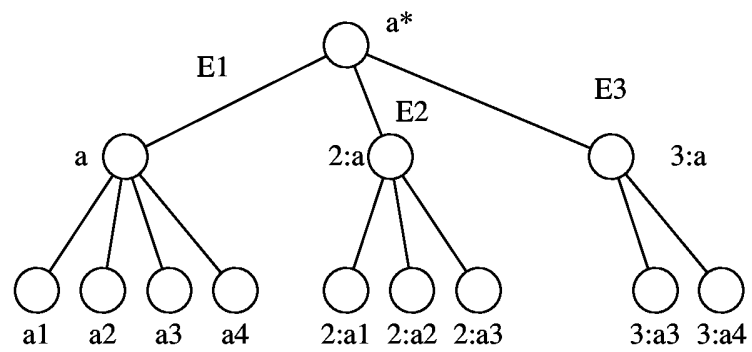
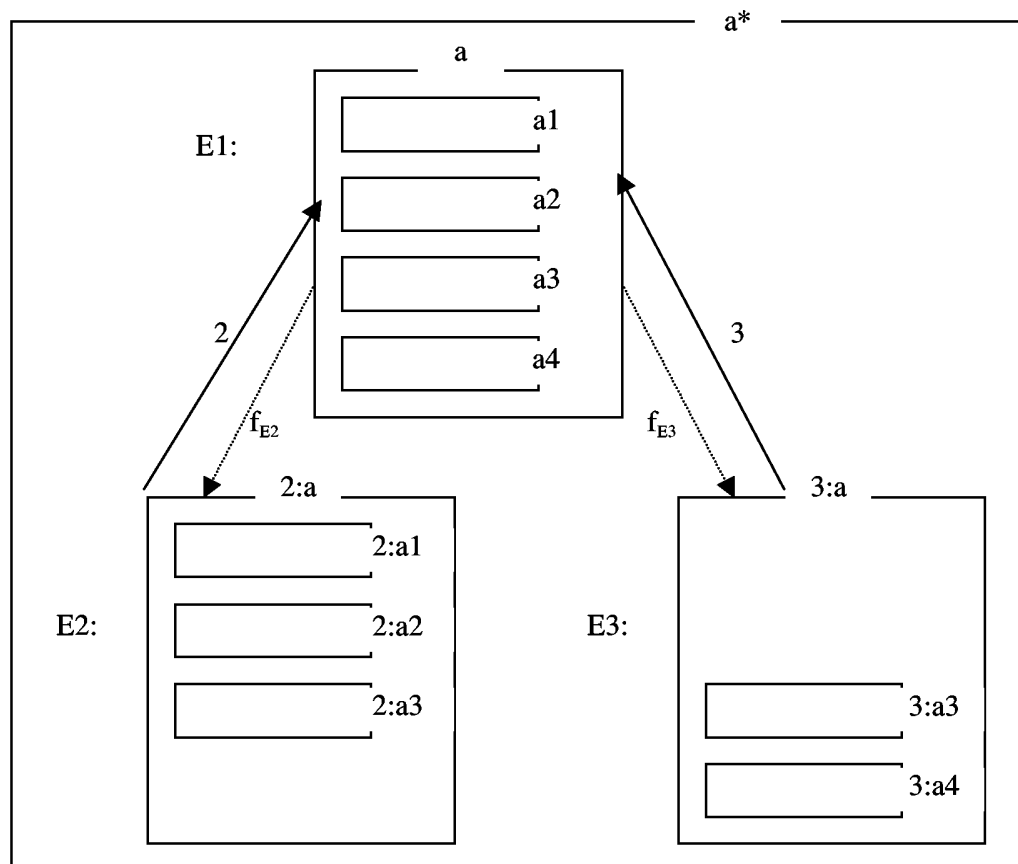


Figure 129



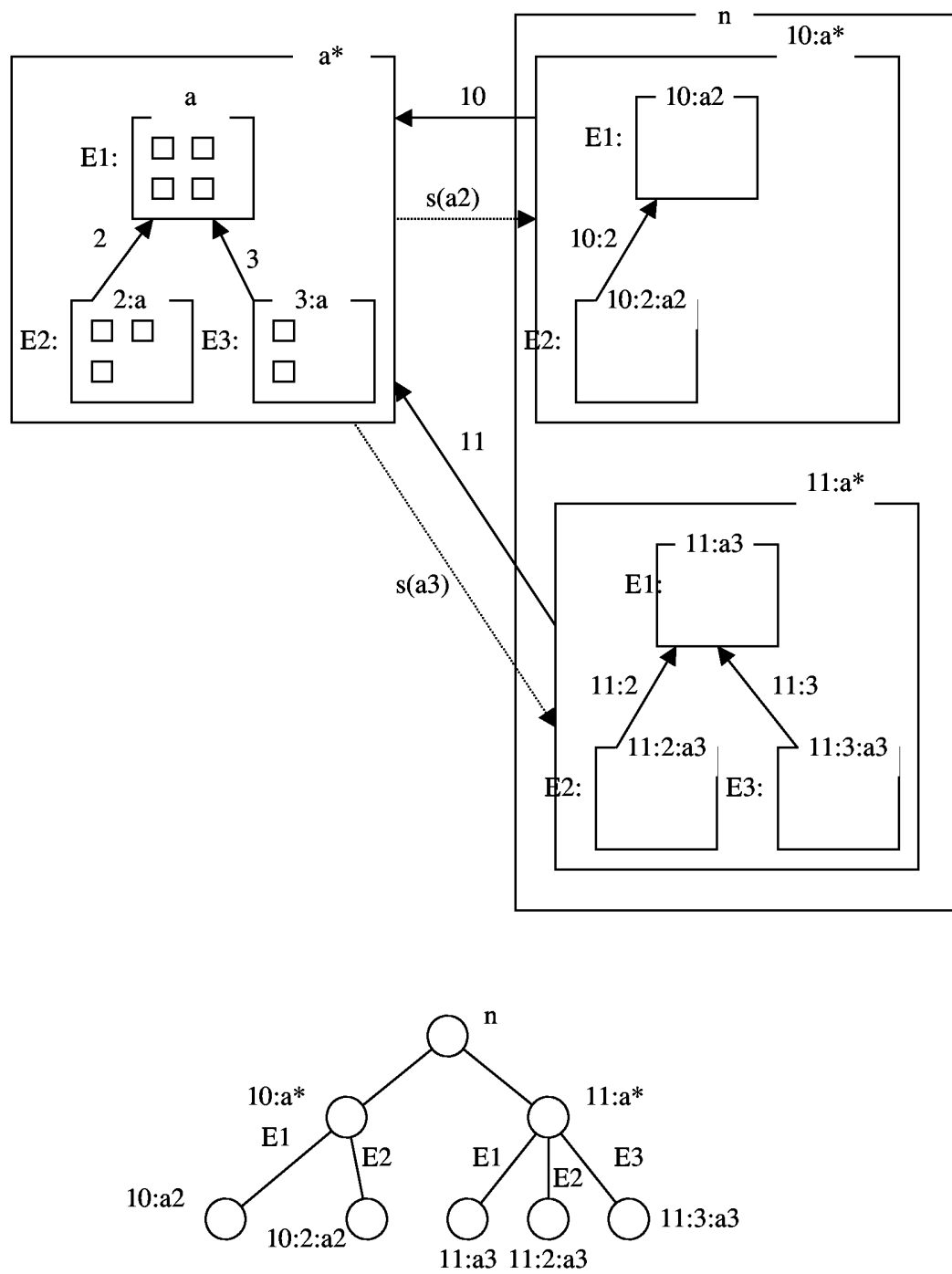


Figure 131

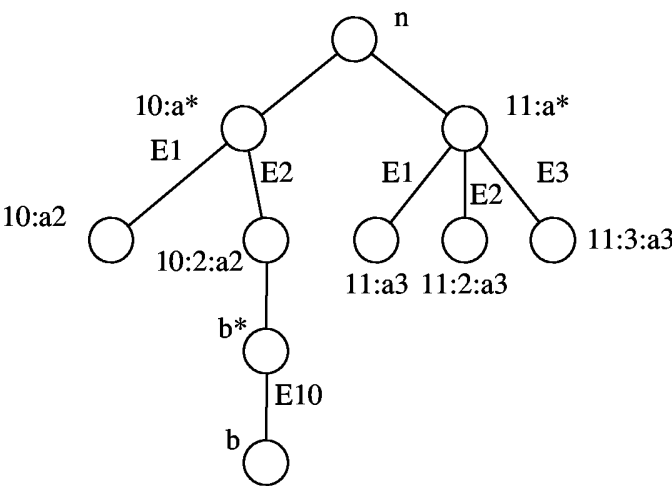


Figure 132

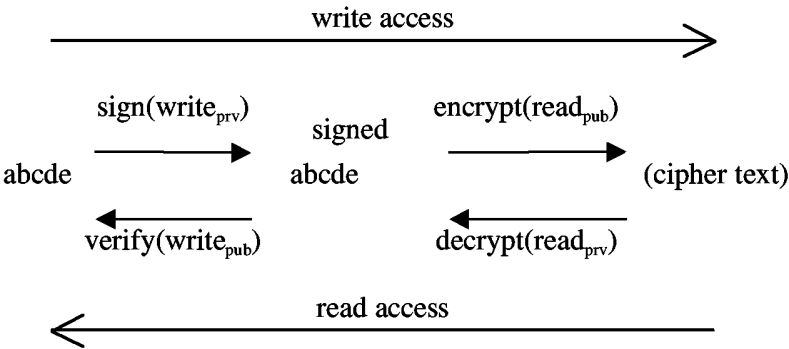


Figure 133

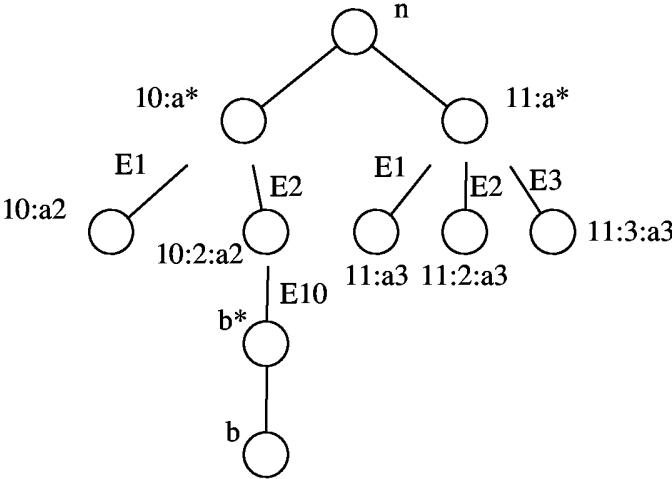


Figure 134

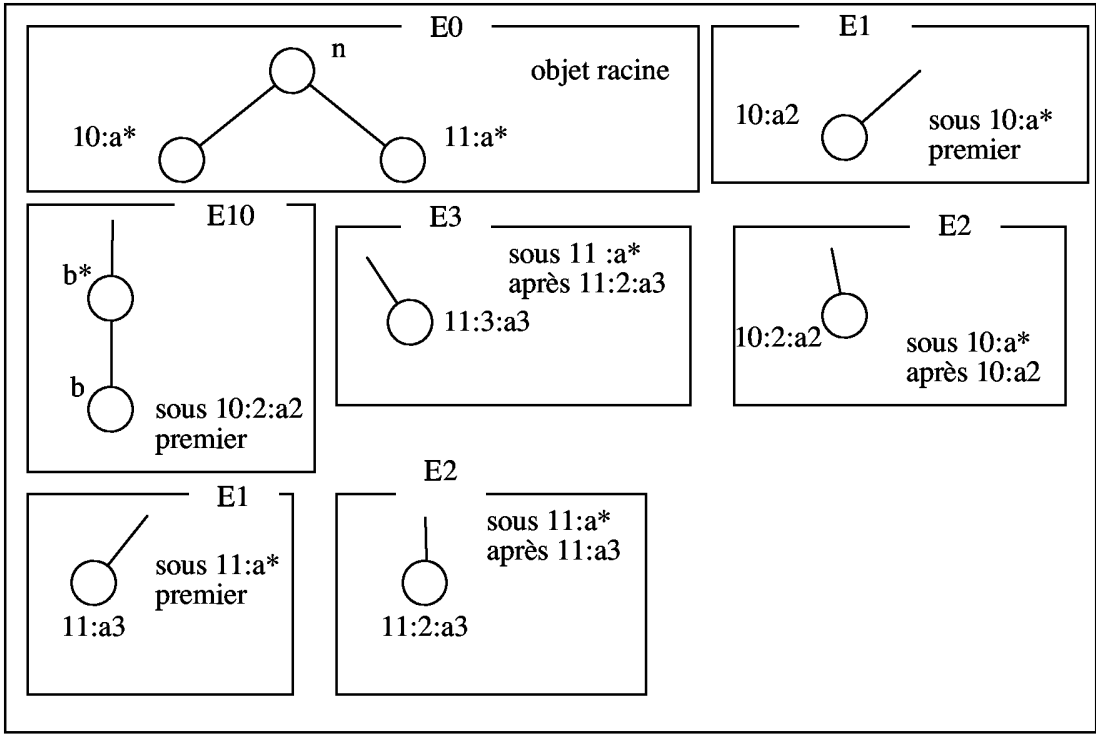


Figure 135

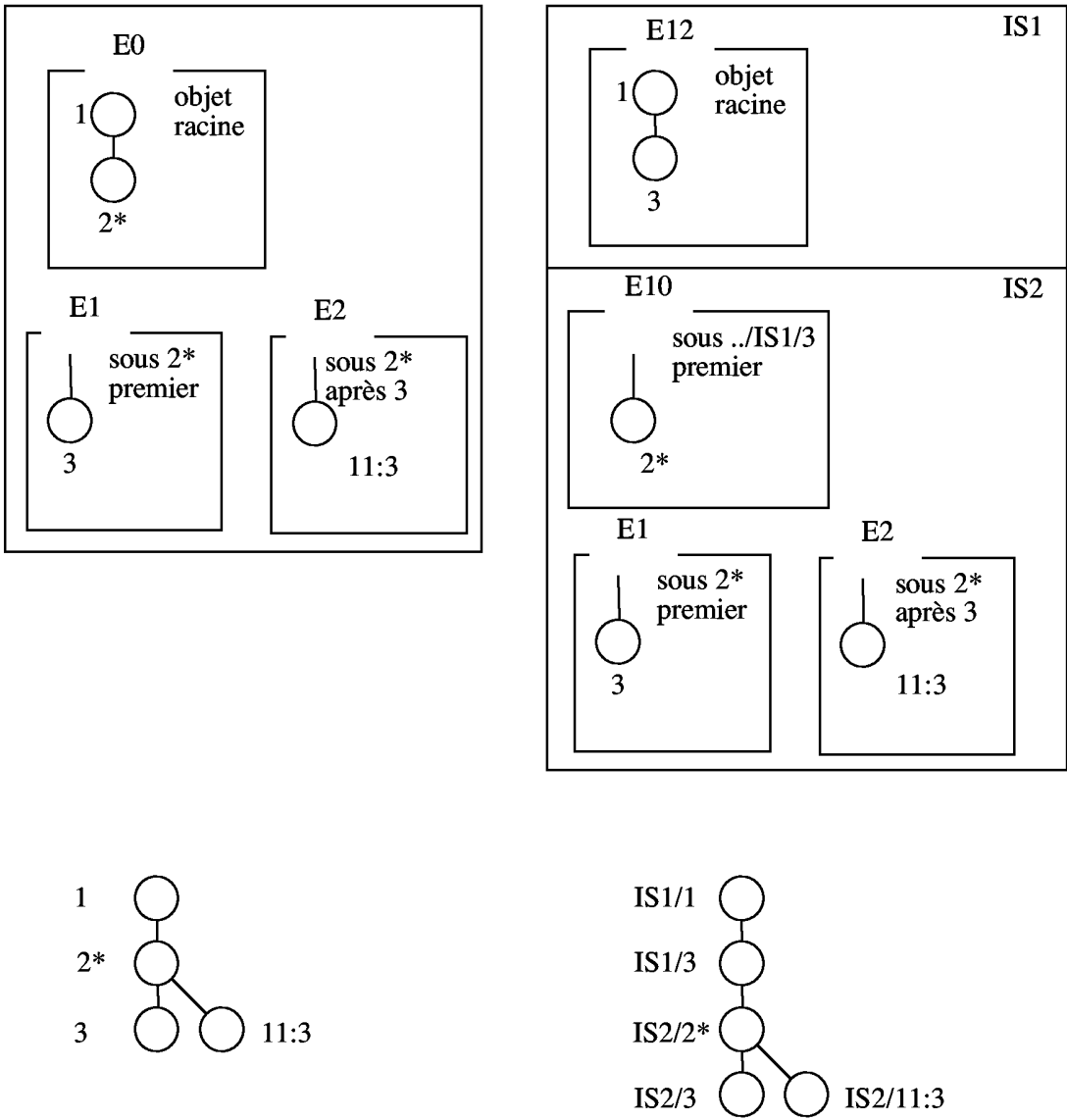


Figure 136

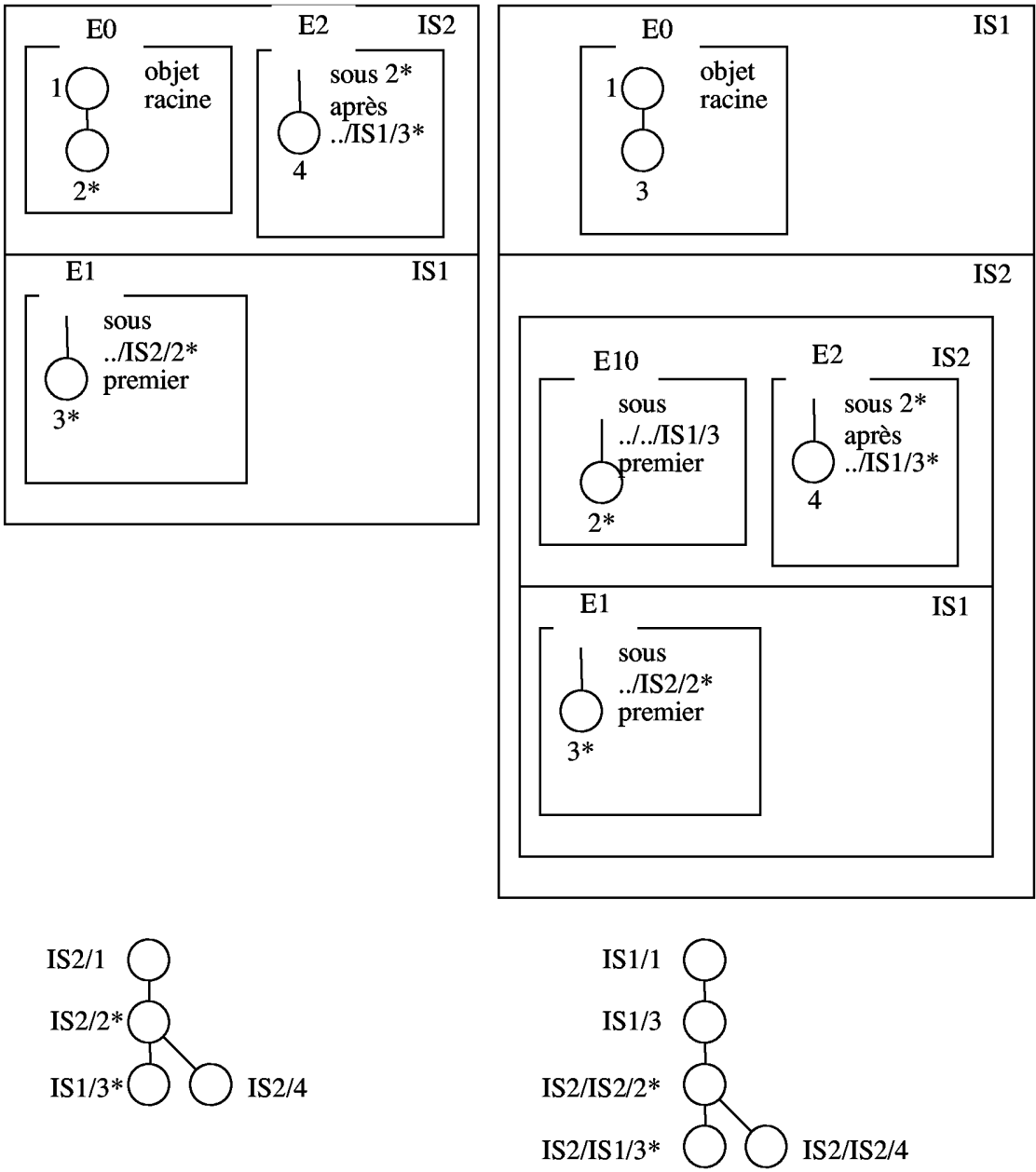


Figure 137

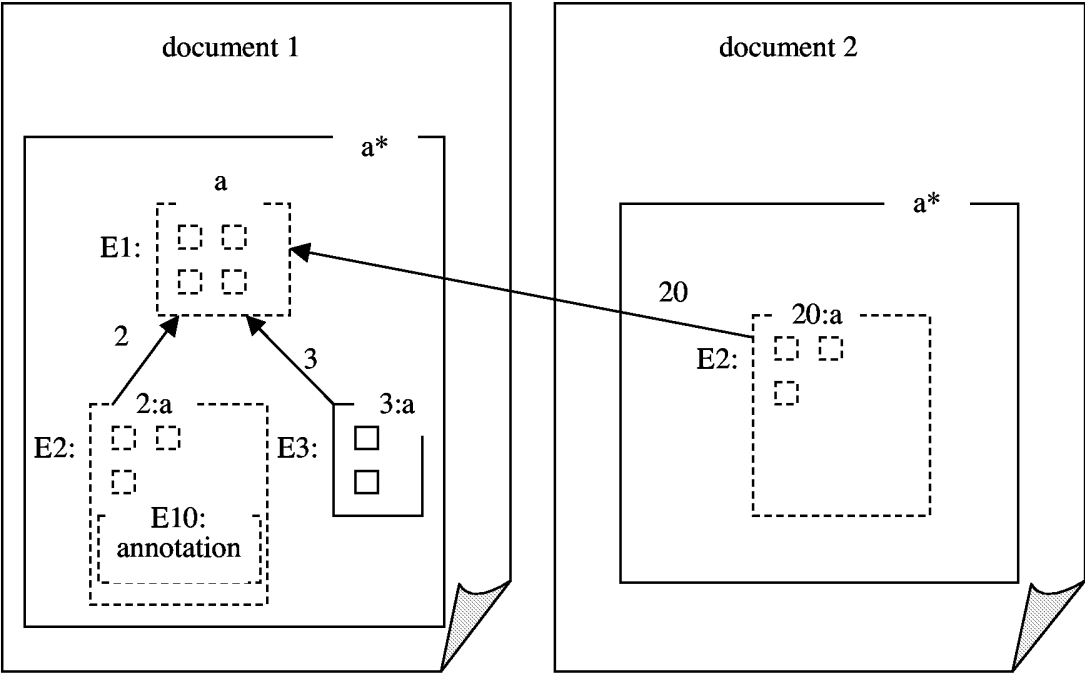


Figure 138

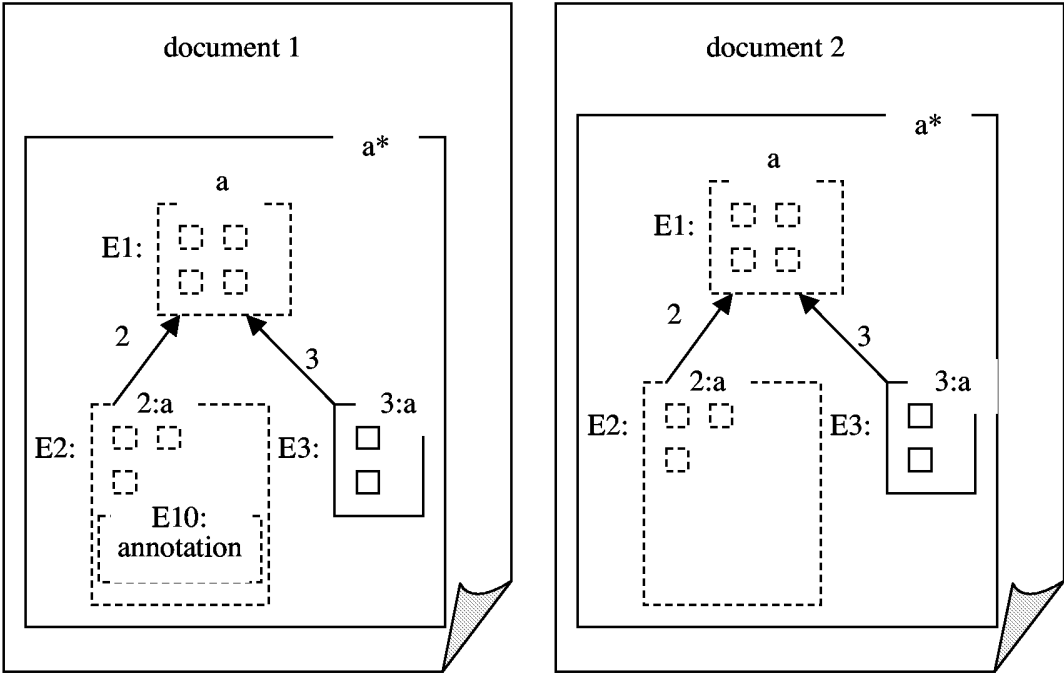


Figure 139

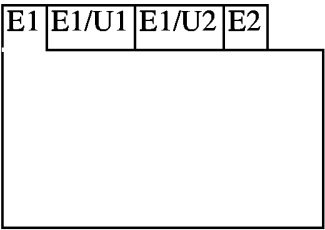


Figure 140

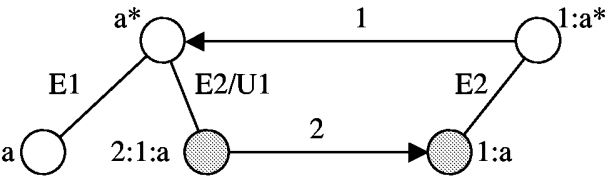


Figure 141

5

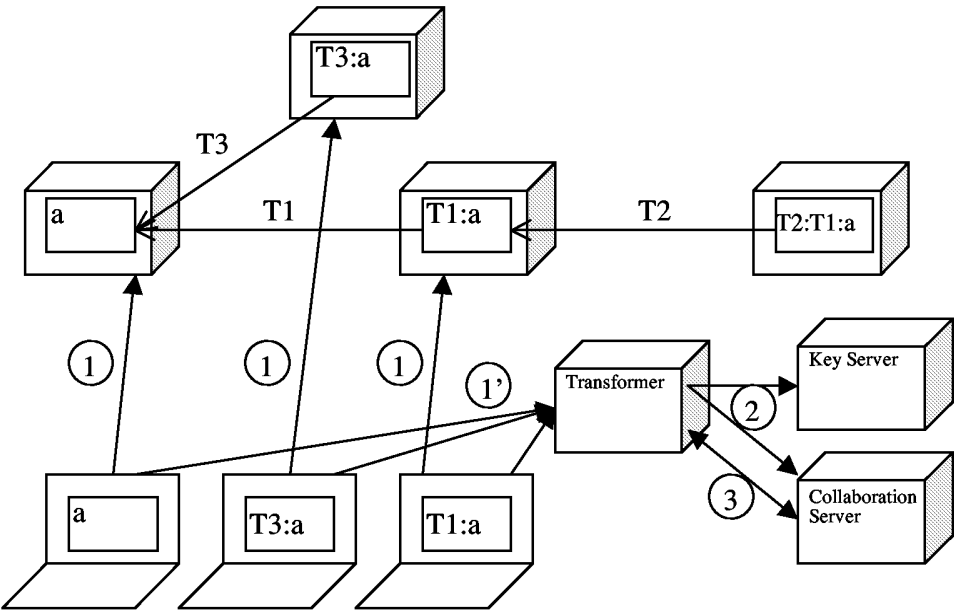


Figure 142

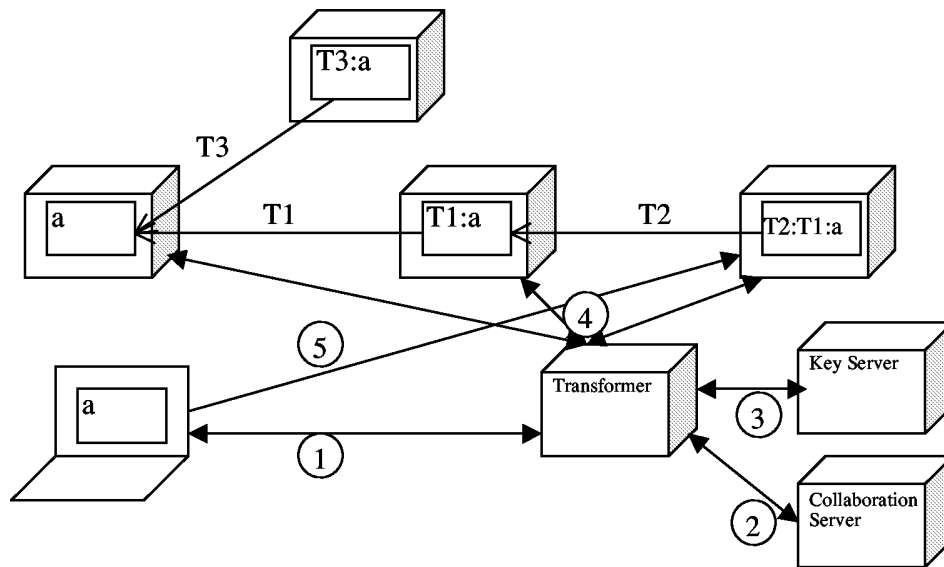


Figure 143