Evolution d'ontologie : Validation des changements basée sur l'évaluation

Rim Djedidi — Hassane Abboute — Marie-Aude Aufaure

Département Informatique, Supélec Campus de Gif
Plateau du Moulon – 3, rue Joliot Curie – 91192 Gif sur Yvette Cedex
{rim.djedidi , hassane.abboute, marie-aude.aufaure}@supelec.fr

RÉSUMÉ. L'évolution d'ontologie est définie comme étant l'adaptation de l'ontologie aux besoins de changements et la propagation de ces changements aux artefacts dépendants tout en préservant la consistance. Par ailleurs, l'évaluation d'ontologie est souvent utilisée pour choisir l'ontologie la plus appropriée en fonction des besoins de l'utilisateur. L'évaluation prend en considération plusieurs aspects de qualité tels que la structure et l'usage. Dans cet article, nous proposons une approche de gestion de l'évolution d'ontologie basée sur la validation de la consistance et l'évaluation de la qualité. La validation de la consistance consiste à vérifier que chaque axiome reste vrai après l'application du changement. L'évaluation de la qualité permet de décider de l'acceptation finale des changements. Un changement qui permet d'améliorer la qualité peut être validé automatiquement sans l'intervention de l'expert.

ABSTRACT. Ontology Evolution: Changes validation based on Evaluation. Ontology evolution is defined as the ontology adaptation to the needs of changes and the propagation of these changes to the dependent artefacts while preserving consistency. In addition, ontology evaluation is often used to choose the most suitable ontology according to user needs. Evaluation takes into account several quality aspects such as structure and usage. In this paper, we propose an approach for ontology evolution management based on consistency validation and quality evaluation. Consistency validation consists in checking that each axiom remains true after change application. Quality evaluation helps deciding on the final acceptance of changes. A change which improves the quality can be automatically validated without expert intervention.

MOTS-CLÉS : évolution d'ontologie, gestion de changements, consistance, axiomes, évaluation d'ontologie, qualité d'ontologie,

KEYWORDS: ontology evolution, change management, consistency, axioms, ontology evaluation, ontology quality.

1. Introduction

Les ontologies sont souvent utilisées dans des environnements dynamiques, multi-acteurs et distribués. Elles ne peuvent être pensées comme des produits finis qui restent stables une fois achevés. Les motifs de changements sont multiples, par exemple le domaine de définition peut évoluer, ce qui implique des modifications dans l'ontologie pour rendre compte de cette évolution ; la réutilisation de l'ontologie pour des tâches différentes nécessite également son adaptation ; la conceptualisation de l'ontologie peut être affinée à travers un processus de construction itératif et incrémental.

L'évolution et l'évaluation doivent être considérées comme des parties prenantes du cycle de vie des ontologies pour que celles-ci puissent garder leur intérêt par rapport aux applications pour lesquelles elles ont été construites. En effet, l'évolution des connaissances d'une ontologie a une influence sur l'évolution et la maintenance des systèmes utilisant cette ontologie (Orme *et al.*, 2006). De plus, la qualité et la validité des connaissances modélisées par l'ontologie affectent directement la qualité et la validité de ces systèmes.

Dans cet article[1], nous proposons une approche de gestion d'évolution d'ontologies basée sur la vérification des axiomes et l'évaluation de la qualité. Après application d'un changement, la consistance de l'ontologie est d'abord vérifiée. Ensuite, la qualité de l'ontologie est évaluée. Si la qualité est améliorée (ou préservée), alors le changement peut être validé directement. Dans le cas contraire, l'expert interviendra pour juger le changement. Cette méthodologie permet ainsi de minimiser la dépendance vis-à-vis de l'expert et d'automatiser la phase de validation de changement.

Cet article est structuré comme suit : dans les sections 2 et 3, nous présentons une synthèse des principales approches concernant respectivement, l'évolution et l'évaluation d'ontologie. Dans la section 4, nous décrivons l'approche d'évolution d'ontologie que nous proposons. Un premier prototype de l'approche est décrit dans la section 5. La dernière section synthétise les différents points de l'approche proposée et présente nos travaux futurs.

2. Evolution d'ontologie

L'évolution d'une ontologie est l'adaptation aux changements et la propagation de ces changements au niveau des artefacts dépendants, c'est-à-dire les objets référencés par l'ontologie ainsi que les ontologies et les applications qui lui sont liées (Sure *et al*, 2004). Les changements sont classés selon trois niveaux (Haase *et al*, 2004) : des changements du domaine modélisé, des changements de conceptualisation et des changements de spécification. Le domaine est une partie du

1. Ce travail est financé par l'Agence National de Recherche dans le cadre du Projet-ANR DAFOE.

monde réel, il est donc dynamique et évolue dans le temps. La conceptualisation peut changer en raison d'une nouvelle observation ou d'une restructuration des connaissances. Les changements de spécification se rapportent à la description formelle de l'ontologie (le langage de représentation). Les changements sont aussi classés en deux catégories (Xuan *et al*, 2006): évolution normale et révolution. La première permet d'enrichir l'ontologie par de nouvelles connaissances sans remettre en cause celles qui existent déjà, c'est le principe de « continuité ontologique ». La révolution par contre, permet d'infirmer des axiomes vrais.

Plusieurs approches d'évolution ont été proposées dans la littérature. (Xuan *et al*, 2006) proposent un modèle de gestion de versions pour les entrepôts basés-ontologies, à travers un processus d'évolution asynchrone basé sur le principe de « continuité ontologique » et contraint par la permanence des classes, des propriétés et des subsomptions et par la description des instances.

(Flouris *et al*, 2005) adaptent les principes de changements de croyance (*Belief Changes*) à l'évolution d'ontologies. Ils identifient quatre opérations de changements : *Révision* et *contraction* pour les changements liés à la conceptualisation (c'est la perception du domaine qui change et non le domaine lui-même) et *mise à jour* et *suppression* pour les changements du domaine (nouvelles réalités). Les ontologies sont représentées sous forme d'axiomes. Les changements sont déclenchés par de nouveaux faits (opérandes de changement) introduits au système et sont appliqués à travers une séquence de modifications générées automatiquement. Les changements sont exprimés sous forme d'un ensemble de propositions d'axiomes conformément au modèle de logique de description dans lequel les axiomes de l'ontologie sont exprimés.

La méthodologie *Boemie* (Castano *et al*, 2006) exploite les résultats d'extraction d'information pour assurer le peuplement, l'enrichissement et la coordination d'ontologies multimédia. L'ontologie évoluée est ensuite utilisée pour améliorer le module d'extraction à travers un processus cyclique. *Boemie* se caractérise par l'utilisation de multiple modules ontologiques, chacun assurant un objectif indépendant de structuration de l'espace de connaissances ; par un processus dirigé par des patterns, chaque pattern correspond à une combinaison typique de connaissances de l'ontologie et d'objets multimédia produits par le processus d'extraction ; par une approche ouverte aux sources externes pour l'identification de nouvelles connaissances ; et par la minimisation de la participation humaine à travers des services de support.

(Stojanovic *et al*, 2002) organisent le processus d'évolution en six phases : détection du changement, représentation du changement, sémantique du changement, implémentation du changement, propagation du changement et validation du changement. Certaines de ces phases sont aussi retrouvées dans d'autres approches d'évolution.

2.1. *Détection du changement*

La détection des changements se base sur l'application de méthodes heuristiques et sur les besoins explicites générés par les développeurs d'ontologies et les utilisateurs, et les besoins implicites reflétés par le comportement du système. La découverte des changements peut être dirigée par (Stojanovic *et al,* 2004) : la structure de l'ontologie (*structure-driven*), l'usage résultant des modèles d'utilisation (*usage-driven*) et les données relatives à l'évolution des sources (*data-driven*).

2.2. *Représentation du changement*

Une pratique courante consiste à définir une taxonomie ou une ontologie de changements pour le modèle d'ontologie. (Stojanovic *et al,* 2004) ont définit une taxonomie des changements relatifs au modèle KAON. Trois types de changements sont distingués : les changements élémentaires, les changements composés qui modifient un seul niveau de voisinage des entités de l'ontologie et les changements complexes qui peuvent être décomposés en plusieurs changements composés.

(Klein *et al,* 2003) présentent une taxonomie des opérations de changements selon deux dimensions : atomique vs composée et simple vs riche. Les opérations atomiques sont les opérations qui ne peuvent être divisées en opérations plus élémentaires tandis que les opérations composées fournissent un mécanisme de regroupement d'opérations concernant une entité logique. Les changements simples peuvent être détectés en analysant seulement la structure de l'ontologie, tandis que les changements riches incorporent des informations sur l'implication d'une opération sur le modèle logique de l'ontologie.

Si l'on compare les deux travaux, les changements élémentaires et complexes dans (Stojanovic *et al,* 2004) correspondent aux changements atomiques et composés dans (Klein *et al,* 2003). Dans (Stojanovic *et al,* 2004), il n'y a pas de distinction entre les changements simples et les changements riches. Ces auteurs ont présenté une ontologie pour modéliser les changements d'ontologies et les opérations associées.

2.3. *Sémantique du changement*

La sémantique du changement est la phase la plus cruciale du processus d'évolution car les effets directs et indirects des changements sont traités. Ainsi, la suppression d'un concept nécessite de prendre une décision concernant les instances de ce concept (par exemple, les supprimer ou les reclasser). (Stojanovic *et al,* 2002) suggèrent que la décision finale revienne indirectement au développeur d'ontologie à travers le choix de certaines stratégies prédéterminées. La sémantique du

changement se rapporte aux effets du changement sur l'ontologie elle-même, en particulier la vérification et la maintenance de la consistance. La signification de la consistance dépend beaucoup du modèle fondamental de l'ontologie. (Stojanovic *et al,* 2004) définissent la consistance comme suit : une ontologie simple O est dite consistante et respecte son modèle si et seulement si, elle préserve les contraintes définies pour le modèle fondamental de l'ontologie. Les auteurs décrivent et comparent deux approches pour vérifier la consistance d'une ontologie : la *vérification à posteriori* où les changements sont appliqués en premier, ensuite on vérifie si l'ontologie mise à jour satisfait les contraintes de consistance et la *vérification à priori* qui définit un ensemble de conditions pour chaque changement. Dans ce cas, la consistance sera maintenue, si l'ontologie est déjà conforme avant la mise à jour et si les conditions sont satisfaites.

2.4. *Implémentation du changement*

L**'**implémentation du changement ou l'application physique du changement à l'ontologie, consiste à informer le développeur d'ontologie de toutes les conséquences d'une demande de changement (notification du changement), à appliquer tous les changements nécessaires et dérivés (application du changement) et à garder trace des changements effectués (notation de tous les changements).

2.5. *Propagation du changement*

La mise à jour d'une ontologie peut avoir une influence négative sur d'autres ontologies dépendantes. La phase de propagation du changement consiste à assurer la consistance des objets dépendants après qu'une mise à jour de l'ontologie ait été effectuée. Ces objets peuvent inclure des ontologies et des applications fonctionnant avec l'ontologie. (Maedche *et al,* 2003) présentent deux méthodes de propagation de changements. La première consiste à propager le changement aux éléments dépendants dès son application à l'ontologie. Par contre, la deuxième méthode consiste à propager les changements seulement à la demande explicite de chacun des éléments dépendants. Les auteurs ont favorisé la première méthode et ont présenté une approche d'évolution dans le contexte d'ontologies distribuées et dépendantes. Ils définissent la notion de consistance des ontologies dépendantes : une ontologie dépendante est dite consistante si l'ontologie elle-même et toutes les ontologies incluses, observées seules et indépendamment des ontologies dans lesquelles elles sont réutilisées, sont des ontologies simples et consistantes. Les auteurs définissent également la notion de consistance pour le cas des ontologies multiples : une ontologie est dite consistante si son originale et toutes les ontologies incluses sont consistantes.

2.6. *Validation du changement*

Dans cette phase, le développeur d'ontologie doit passer en revue l'ensemble des changements et les défaire si besoin. Les changements peuvent être annulés pour différentes raisons : le développeur d'ontologie ne comprend pas l'impact réel du changement et n'approuve pas ce changement, le changement est effectué pour des buts expérimentaux, ou dans un contexte collaboratif, les développeurs d'ontologies peuvent avoir des avis différents sur la façon avec laquelle l'ontologie devrait être modifiée. La validation permet de justifier les changements effectués. Au niveau de cette phase, d'autres problèmes peuvent être identifiés ce qui nécessite de reprendre les phases d'évolution selon un processus cyclique.

3. Evaluation d'ontologie

L'évaluation des ontologies est une question primordiale notamment pour l'utilisation des ontologies dans le contexte du web sémantique et par d'autres applications basées sur la sémantique. Souvent, les utilisateurs doivent choisir, parmi une multitude d'ontologies, l'ontologie la plus appropriée en fonction de leurs besoins. Ce choix se base essentiellement, sur le résultat de l'évaluation de chacune des ontologies. Par ailleurs, l'évaluation peut servir de guide lors du processus de construction et dans toute étape de raffinement d'ontologie. (Haase *et al*, 2005) ont intégré l'évaluation dans le processus d'évolution pour détecter les changements utiles à l'ontologie.

Plusieurs approches d'évaluation d'ontologies ont été proposées dans la littérature (Brank *et al*, 2005) (Supekar *et al*, 2005) (Yang *et al*, 2006). Certaines approches se basent sur l'utilisation de l'ontologie dans une application et l'évaluation des résultats de cette utilisation, d'autres se basent sur une comparaison avec les sources de données (collection de documents) mais dans ce cas l'évaluation est faite par les humains en prenant en considération plusieurs critères, normes et conditions.

Une autre classification des approches d'évaluation peut se faire en fonction du niveau d'évaluation. L'ontologie étant une structure assez complexe, il est plus pratique de procéder par une évaluation par niveau plutôt que d'évaluer l'ontologie dans son ensemble. Cette approche est d'autant plus intéressante dans le cas où l'évaluation doit être automatisée. Différents niveaux ont été définis dans la littérature, mais les diverses définitions s'accordent sur les niveaux suivants : aspect lexical/vocabulaire, taxonomie, application, données, et approche multicritères.

Le niveau lexical se base sur les concepts, les instances, les faits décrits dans l'ontologie et le vocabulaire utilisé pour les représenter. L'évaluation à ce niveau tend à effectuer des comparaisons avec diverses sources de données concernant le domaine (corpus de textes spécifique au domaine). (Maedche *et al*, 2002) ont proposé une approche pour évaluer le niveau lexical/vocabulaire d'une ontologie en

calculant la similitude entre deux termes avec la distance de *Levenshtein* (similarité lexicale). Le contenu lexical d'une ontologie peut être évalué en utilisant la précision et le rappel souvent utilisés dans le traitement automatique des langues. Dans ce contexte, la précision représente le pourcentage des termes lexicaux de l'ontologie (chaînes utilisées pour nommer les concepts) qui apparaissent dans le standard par rapport au nombre total des mots de l'ontologie. Le rappel représente le pourcentage des termes lexicaux du standard qui apparaissant également comme noms de concepts dans l'ontologie par rapport au nombre total de termes lexicaux du standard. Cette approche a également été utilisée pour évaluer le lexique d'une ontologie à d'autres niveaux, par exemple les chaînes utilisées pour identifier les relations.

En ce qui concerne la taxonomie, (Gangemi *et al*, 2005) ont défini des mesures structurelles impliquant la profondeur, la largeur, la distribution des feuilles, la densité, la modularité, la consistance, la complexité, etc. (Maedche *et al*, 2002) ont proposé plusieurs mesures pour comparer les aspects relationnels entre deux ontologies. D'un autre coté, (Brewster *et al*, 2004) ont proposé une approche basée sur les sources de données pour évaluer le degré d'adaptation de la structure de l'ontologie à un corpus de documents.

Typiquement, une ontologie est utilisée par des applications. Le résultat de l'application dépend d'une certaine façon de l'ontologie utilisée. Ainsi, pour pouvoir porter un jugement sur l'ontologie, il est fortement intéressant d'évaluer le résultat de l'utilisation de l'application pour certaines tâches. Cette approche présente quelques inconvénients (Porzel *et al*, 2004) : (1) le résultat se base sur une tâche particulière et il est difficile de généraliser cette observation ; (2) l'ontologie constitue un petit composant de l'application et son impact sur le résultat peut être faible et indirect ; (3) la comparaison de plusieurs ontologies ne peut se faire que si elles sont utilisées par la même application.

L'évaluation basée sur les données consiste à comparer l'ontologie à une collection de documents décrivant le domaine d'intérêt. (Patel *et al*, 2004) ont montré comment déterminer si une ontologie se rapporte à un domaine d'intérêt particulier, et classifient les ontologies dans un annuaire de domaine d'intérêt. Des données textuelles telles que les noms de concepts, sont extraites à partir de l'ontologie et utilisées comme entrée d'un modèle de classification. De même, (Brewster *et al*, 2004) ont proposé une approche d'évaluation basée sur les corpus. Ainsi, plutôt que de comparer plusieurs ontologies entre elles afin de choisir la plus appropriée à un usage particulier, les auteurs proposent de comparer les ontologies à un corpus. Après avoir procédé à l'extraction automatique de termes à partir du corpus, le chevauchement entre les termes obtenus par extraction et les termes de l'ontologie sera calculé. Une ontologie peut être pénalisée par les termes disponibles au niveau du corpus et absents dans l'ontologie, comme elle peut être pénalisée par les termes disponibles au niveau de l'ontologie et absents dans le corpus.

L'approche d'évaluation multicritères est souvent utilisée pour la sélection de l'ontologie la plus appropriée parmi un ensemble d'ontologies ce qui se ramène à une problématique de prise de décision. Cette approche est basée sur la définition d'un ensemble de critères (attributs). Pour chaque critère, l'ontologie est évaluée et un score est attribué. De plus, un poids est assigné à chaque critère. (Burton-Jones *et al.,* 2004) ont proposé une approche avec 10 critères : l'éligibilité (fréquence des erreurs syntaxiques), la richesse, l'interprétabilité (la présence des termes utilisés dans WordNet), la consistance (nombre de concepts impliqués dans des contradictions), la clarté (les termes utilisés dans l'ontologie ont-ils plusieurs sens dans WordNet?), la compréhensivité, l'exactitude (pourcentage de fausses relations), la pertinence, l'autorité (combien d'autres ontologies utilisent les concepts de l'ontologie à évaluer?), l'historique (nombre d'accès à l'ontologie). (Fox *et al,* 1998) ont proposé un autre ensemble de critères : la complétude fonctionnelle (l'ontologie contient-elle assez d'information?), la généralité (l'ontologie est-elle assez générale pour qu'elle soit partagée par plusieurs utilisateurs?), l'efficacité du raisonnement supporté par l'ontologie, la compréhension, la précision/granularité (l'ontologie supporte-t-elle plusieurs niveaux d'abstraction/détail?), la minimalité (l'ontologie contient-elle tous les concepts nécessaires?). (Lozano-Tello *et al,* 2004) ont proposé 117 critères. Ces critères couvrent plusieurs aspects : le contenu de l'ontologie (concepts, relations, taxonomie, axiomes), la méthodologie utilisée lors de la construction de l'ontologie, le coût d'utilisation de l'ontologie et les outils disponibles.

4. Une approche d'évolution d'ontologie basée sur l'évaluation

L'évolution d'ontologie fait émerger un certain nombre de questions dont : L'application d'un changement n'a-t-elle pas d'impacts négatifs sur la consistance de l'ontologie ? Le changement permet-il d'améliorer la qualité de l'ontologie ? Pour traiter ces questions, la validation de la consistance et l'évaluation de la qualité doivent être intégrées au processus d'évolution d'ontologies.

Tout changement peut mener à des contradictions au sein de l'ontologie. Il est donc indispensable de vérifier la consistance de l'ontologie. Un changement menant à une inconsistance ne peut être validé. Lorsqu'un changement (n'ayant pas d'impact négatif sur la consistance de l'ontologie) est validé, il serait intéressant de procéder à l'évaluation de l'ontologie afin de mesurer l'impact du changement sur la qualité de l'ontologie. L'évaluation peut en effet, offrir un moyen de validation des changements utiles qui vont permettre d'améliorer la qualité de l'ontologie et ainsi, de rendre automatique la gestion de l'évolution en étant moins dépendant de l'expert du domaine. Ainsi, tout changement qui n'affecte pas la consistance et qui permet de préserver ou d'améliorer la qualité de l'ontologie sera intégré.

L'approche d'évolution proposée est organisée en six phases : identification du changement, représentation du changement, implémentation du changement, validation du changement, évaluation de la qualité de l'ontologie et annotation.

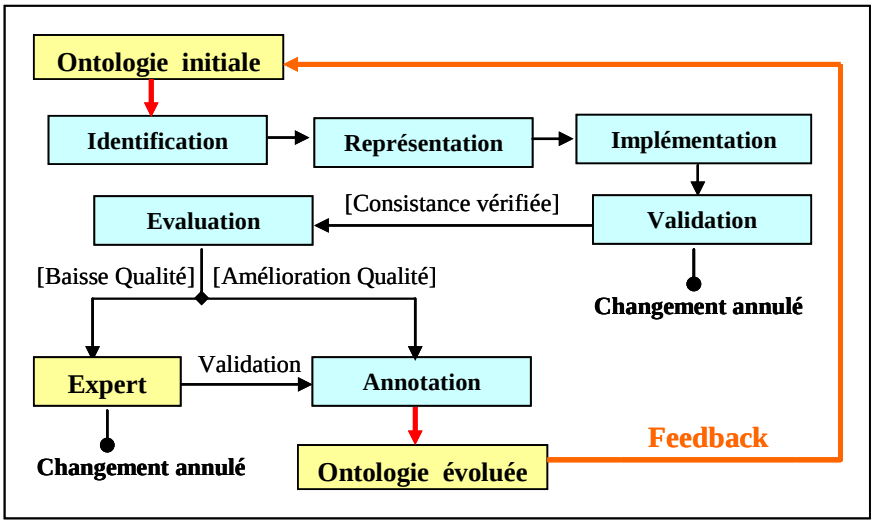

Figure 1. *Approche d'évolution d'ontologie.*

Les changements à apporter à l'ontologie peuvent être identifiés soit de manière descendante à partir des besoins exprimés par son utilisation ou par de nouvelles connaissances du domaine, soit de manière ascendante en analysant l'ontologie elle-même. Le changement identifié est ensuite transcrit en un format approprié au modèle de l'ontologie puis implémenté physiquement. La phase d'implémentation consiste à appliquer le changement identifié et ses changements dérivés en propageant automatiquement le changement vers les entités qui en dépendent. L'application du changement se fait tout en préservant la trace des modifications effectuées et les métadonnées associées. A l'issue de cette phase, l'ontologie sera « provisoirement » mise à jour. L'intégration finale du changement est contrainte par la vérification de la consistance et par la préservation de la qualité de l'ontologie. Ainsi, nous proposons de valider les axiomes de l'ontologie en appliquant le principe de continuité ontologique et d'employer les techniques d'évaluation pour mesurer la qualité de l'ontologie. Une fois les changements acceptés, la phase d'annotation permet de garder la trace des opérations de changements et l'historique de l'évolution de l'ontologie.

Les phases de validation et d'évaluation représentent le noyau de l'approche proposée et sont détaillées dans les sections suivantes.

4.1. *Validation du changement*

La phase de validation consiste à vérifier la consistance de l'ontologie. L'inconsistance peut être reflétée par le fait qu'un axiome de l'ontologie ne soit plus

valide après l'application du changement. Les axiomes sont composés de schémas d'axiomes, propriétés conceptuelles enrichissant les concepts et les relations, et des axiomes de domaines (Fürst *et al*, 2005). Comme les axiomes de domaine dépendent des contraintes liées au domaine modélisé et que notre objectif est d'établir une approche de validation de changements automatique et indépendante du contexte d'utilisation et du domaine, la validation sera basée uniquement sur les schémas d'axiomes. Un schéma d'axiome permet de définir :

– La relation *Is-a* entre deux concepts ou deux relations.

– La propriété d'abstraction d'un concept.

– La propriété de disjonction entre deux concepts.

– La signature d'une relation qui permet de préciser les concepts.

– Les propriétés algébriques d'une relation.

– L'exclusivité de deux relations.

– L'incompatibilité de deux relations.

– Les cardinalités minimale et maximale d'une relation.

Nous avons défini un algorithme de vérification pour chaque axiome.

Disjonction de concepts : Deux concepts C_i et C_j sont dits disjoints s'ils n'ont pas d'instances communes. L'idée de base de cet algorithme consiste à retrouver l'ensemble des instances de C_i et l'ensemble des instances de C_j, et vérifier si l'intersection des deux ensembles est un ensemble vide. Ins(C) : les instances du concept C. Non-Feuille(C)=0 si le concept C est une feuille, Sinon 1. Fils(C) : l'ensemble des concepts fils de C (Fils(C) = {C_1, C_2,….,C_k}). Fils(C)[n] : le $n^{ième}$ concept fils de C. Nombre-fils(C) : le nombre de concepts fils de C. Ins : est un ensemble d'instances initialisé à ensemble vide. Instance(C) est une procédure récursive qui permet de calculer les instances du concept C et celles de ses descendants.

```
Instance(C)
Begin
 Ins = Ins ∪ Ins(C)
 If Non-Feuille(C) Then
  For n=1 à Nombre-fils(C)
  Do {
   C = Fils(C) [n]
   Instance(C)}
End
```

```
Disjonction (Ci,Cj)
Begin
 If (Instance(Ci) ∩
    Instance(Cj) == ∅)
 Then
    Disjonction préservée
 Else
    Disjonction non
    préservée
End
```

Tableau 1. *Algorithme de disjonction.*

Incompatibilité de deux relations : Deux relations n-aires R_i et R_j ayant la même signature (C_1, C_2,…, C_n) sont incompatibles si elles ne peuvent lier le même ensemble d'instances. Ins_i = Instances (C_1, C_2,…, C_n, R_i) : l'ensemble des k n-uplets d'instances reliées par la relation R_i. Par exemple : Instances (C_1, C_2,…, C_n, R_i) = {(I_{11}, I_{21},…,I_{n1}), (I_{12}, I_{22},…,I_{n2}),…, (I_{1k}, I_{2k},…,I_{nk})}.

Exclusivité de deux relations : Deux relations n-aires R_i et R_j ayant la même signature (C_1, C_2,…, C_n) sont exclusives si lorsque l'une des relations est niée, l'autre est établie. Autrement dit, tout n-uplets d'instances qui ne vérifie pas l'une des deux relations vérifie l'autre. Ins_i = Instances (C_1, C_2,…, C_n, R_i) : l'ensemble des k n-uplets d'instances reliées par la relation R_i. Ins = Instances-Concepts (C_1, C_2,…, C_n) : l'ensemble de tous les n-uplets d'instances des concepts C_1, C_2,…, C_n.

```
Incompatibilité (Ri,Rj)
Begin
 If (Insi ∩ Insj == ∅)Then
  Incompatibilité
  préservée
 Else
  Incompatibilité non
  préservée
End
```

```
Exclusivité (Ri,Rj)
Begin
 If (Insi ∩ Insj) == ∅)
 Then
  If (Insi ∪Insj == Ins)
  Then
    Exclusivité
    préservée
  Else
    Exclusivité non
    préservée
End
```

Tableau 2. *Algorithmes d'incompatibilité et d'exclusivité de deux relations.*

Symétrie d'une relation binaire : Une relation binaire R entre deux concepts C_i et C_j est dite symétrique si pour tout couple d'instances (Ins_i, Ins_j) qui vérifie la relation R, le couple d'instances (Ins_j, Ins_i) vérifie également cette relation. Instances (C_i, C_j, R) : l'ensemble des couples d'instances qui vérifient la relation R. Cardinalité [Instances (C_i, C_j, R)] : le nombre de couples d'instances qui vérifient la relation R. Instances (C_i, C_j, R)[n] : le $n^{ième}$ couple d'instances qui vérifient la relation R. Appartenir ((Ins_{jn}, Ins_{in}), Instances (C_i, C_j, R)) = 1 si le couple d'instance (Ins_{jn}, Ins_{in}) appartient à l'ensemble Instances (C_i, C_j, R), 0 dans le cas contraire.

Antisymétrie d'une relation binaire : Une relation binaire R entre deux concepts C_i et C_j est dite antisymétrique si pour tout couple d'instances (Ins_i, Ins_j) qui vérifie la relation R, le couple d'instances (Ins_j, Ins_i) ne vérifie pas cette relation.

```
Symétrie (R)
Begin
 Trouv = True
 n = 1
 While(n ≤ Cardinalité
  [Instances (Ci,Cj,R)])et
  (Trouv == True))
  {(Ini,Inj) =
    Instances (Ci,Cj,R)[n]
    If (Appartenir(
     (Inj,Ini),
     Instances(Ci,Cj,R)
     /(Ini,Inj))== 0)
    Then
     Trouv = False
    Else
     n = n + 1 }
 If (Trouv == True) Then
   Symétrie préservée
 Else
   Symétrie non préservée
End
```

```
Antisymétrie (R)
Begin
 Trouv = False
 n = 1
 While(n ≤ Cardinalité
  [Instances (Ci,Cj,R)])et
  (Trouv == False))
  {(Ini,Inj) =
    Instances (Ci,Cj,R)[n]
    If (Appartenir(
     (Inj,Ini),
     Instances(Ci,Cj,R)
     /(Ini,Inj))== 1)
    Then
     Trouv = True
    Else
     n = n + 1 }
 If (Trouv == False) Then
  Antisymétrie préservée
 Else
  Antisymétrie non
  préservée
End
```

Tableau 3. *Algorithmes de symétrie et d'antisymétrie d'une relation binaire.*

Réflexivité d'une relation binaire : Une relation binaire R de signature (C_i, C_j) est dite réflexive si pour toute instance Ins_i du concept C_i, le couple d'instance (Ins_i, Ins_i) vérifie cette relation. Instances (C_i) : l'ensemble des instances du concept C_i. Nombre_Instances (C_i) : le nombre d'instances de C_i. Instances (C_i)[n] : la $n^{ième}$ instance du concept C_i (cf. tableau 4).

Anti-réflexivité d'une relation binaire : Une relation binaire R de signature (C_i, C_j) est dite anti-réflexive si pour toute instance Ins_i du concept C_i, le couple d'instance (Ins_i, Ins_i) ne vérifie pas cette relation (cf. tableau 4).

Transitivité d'une relation binaire : Une relation binaire R entre deux concepts C_1 et C_2 est dite transitive si pour tout couple d'instances (Ins_1, Ins_2) et (Ins_2, Ins_3) qui vérifient la relation R, le couple d'instance (Ins_1, Ins_3) vérifie également cette relation (cf. tableau 5).

Réflexivité (R)

```
Begin
 Trouv = True
 n = 1
 While ((n ≤
  Nombre_Instances (Ci))
  et (Trouv == True))
  {Insin = Instances(Ci)[n]
   If (Appartenir(
   (Insin,Insin),
   Instances(Ci,Cj,R))==0)
   Then
    Trouv = False
   Else
    n = n + 1 }
 If (Trouv == True) Then
   Réflexivité préservée
 Else
   Réflexivité non
   préservée
End
```

Anti-Réflexivité (R)

```
Begin
 Trouv = False
 n = 1
 While ((n ≤
  Nombre_Instances (Ci))
  et (Trouv == False))
  {Insin = Instances(Ci)[n]
   If (Appartenir(
   (Insin,Insin),
   Instances(Ci,Cj,R))==1)
   Then
    Trouv = True
   Else
    n = n + 1 }
 If (Trouv == False) Then
  Anti-réflexivité
  préservée
 Else
  Anti-réflexivité non
  préservée
End
```

Tableau 4. *Algorithmes de réflexivité et d'anti-réflexivité d'une relation binaire.*

Recherche (Ins_j,Instances (C_1,C_2,R))

```
Begin
 k = 1
 Trouv = False
 While ((k ≤ Cardinalité
 [Instances(C1,C2,R)])
 et (Trouv == False))
  {(Insa,Insb) =
   Instances (C1,C2,R)[k]
   If (Insa == Insj) Then
    Trouv = True
   Else
    k = k + 1 }
End
//Recherche = True s'il existe
un couple //d'instances parmi
//Instances (C1,C2,R), qui
//commence par Insj.
```

Transitivité (R)

```
Begin
 n = 1
 Préserve = True
 While ((n ≤ Cardinalité
  [Instances(C1,C2,R)])
  et (Préserve == True))
   {(Insi,Insj) =
    Instances(C1,C2,R)[n]
    Recherche(Insj,
    Instances(C1,C2,R))
    If (Trouv == True)
    Then
     If (Appartenir(
      (Insi,Insb),
      Instances(C1,C2,R))
      == 0)Then
       Préserve = False
       n = n + 1 }
 If (Préserve == True)
 Then
   Transitivité préservée
 Else
  Transitivité non
  préservée
End
```

Tableau 5. Algorithme de transitivité d'une relation binaire.

Cardinalité maximale d'une relation : Une relation n-aire R de signatures (C_1, C_2,…, C_n) a pour cardinalité maximale $Card_{Max}$ si pour toute instance Ins_i de n'importe quel concept $C_i \in (C_1, C_2,\ldots, C_n)$, il existe au plus $Card_{Max}$ (n-1)-uplets (Ins_1, Ins_2,…,Ins_{i-1}, Ins_{i+1},…Ins_n) appartenant à l'ensemble des instances qui vérifient la relation R. Compte (Ins_m, Instances (C_1, C_2,…, C_n, R)) : permet de compter, dans l'ensemble des n-uplets d'instances qui vérifient la relation R, le nombre de n-uplets dans lesquels figure l'instance Ins_m.

Cardinalité minimale d'une relation : Une relation n-aire R de signatures (C_1, C_2,…, C_n) a pour cardinalité minimale $Card_{Min}$ si pour toute instance Ins_i de n'importe quel concept $C_i \in (C_1, C_2,\ldots, C_n)$, il existe au moins $Card_{Min}$ (n-1)-uplets (Ins_1, Ins_2,…,Ins_{i-1}, Ins_{i+1},…Ins_n) appartenant à l'ensemble des instances qui vérifient la relation R.

```
CardMax (R)
Begin
 k = 1
 Préserve = True
 While ((k ≤ n) et
  (Préserve == True))
  {m = 1
   While ((m ≤
   Nombre_Instances(Ck))
   et (Préserve == True))
   {If (Compte(Insm,
   Instances(C1,C2,…,Cn,R))
   > CardMax)
    Then
     Préserve = False
    Else
     m = m + 1 }
   k = k + 1 }
 If (Préserve == True)
 Then
  CardMax respectée
 Else
  CardMax non respectée
End
```

```
CardMin (R)
Begin
k = 1
 Préserve = True
 While ((k ≤ n) et
  (Préserve == True))
  {m = 1
   While ((m ≤
   Nombre_Instances(Ck))
   et (Préserve == True))
   {If (Compte(Insm,
   Instances(C1,C2,…,Cn,R))
   < CardMin)
    Then
     Préserve = False
    Else
     m = m + 1 }
   k = k + 1 }
 If (Préserve == True)
 Then
  CardMin respectée
 Else
  CardMin non respectée
End
```

Tableau 6. *Algorithmes de cardinalité maximale et minimale d'une relation n-aire.*

4.2. *Evaluation de la qualité de l'ontologie*

Une fois la consistance de l'ontologie assurée, la phase d'évaluation permet de mesurer l'impact du changement sur la qualité de l'ontologie. Si le changement mène vers une ontologie de meilleure qualité ou préserve la qualité de l'ontologie alors, il sera directement validé. Dans le cas contraire, l'expert interviendra pour décider du changement.

L'évaluation prend en considération plusieurs aspects de l'ontologie. Le résultat de l'évaluation est comparé à la valeur de l'ancienne évaluation contenue dans l'annotation de l'ontologie (la qualité avant le changement). Si le résultat de la nouvelle évaluation est une note supérieure (ou égale) à celle de l'ancienne évaluation, le changement permettra d'améliorer la qualité de l'ontologie (ou la préserver) et sera donc validé. Dans le cas contraire, le changement influence négativement la qualité de l'ontologie et ne sera pas validé directement mais laissé à la charge de l'expert. Ainsi, l'évaluation intervient comme un moyen permettant de faciliter et guider la validation des changements en prenant en considération la qualité de l'ontologie.

La norme ISO/CEI 9126 décrit les exigences en termes de qualité des produits logiciels. Cette norme permet de mesurer la qualité d'un produit logiciel en prenant en considération plusieurs caractéristiques. (Bansiya *et al*, 2002) ont adapté cette norme pour établir un modèle hiérarchique pour l'évaluation de la qualité d'une conception orientée-objet. Ce modèle comprend une arborescence d'évaluation contenant des caractéristiques, des sous-caractéristiques, des critères et des métriques d'évaluation. Infauditor (Akoka, *et al*, 1996) est une méthodologie pour l'audit de l'ensemble du système d'information d'une entreprise. Elle comprend une arborescence d'évaluation basée sur des domaines, des sous-domaines et des critères d'évaluation. Le même principe d'une arborescence d'évaluation, peut être appliqué pour évaluer la qualité d'une ontologie en prenant en considération plusieurs aspects d'évaluation. Pour chaque aspect, plusieurs critères peuvent être utilisés pour lesquels, différentes métriques d'évaluation peuvent être définies.

Lors de l'évaluation de l'ontologie, deux aspects doivent être principalement pris en considération : l'aspect structurel et l'usage. Pour chaque aspect, plusieurs critères doivent être évalués. Pour la structure, nous avons identifié les critères suivants : la complexité, la cohésion, la modularité, la taxonomie et l'abstraction. Pour l'usage, trois critères ont été évalués : la complétude, la modularité et la compréhension. Il se trouve que certains de ces critères sont parfois opposés les uns aux autres : un changement qui rend l'ontologie plus complexe par exemple, peut améliorer la cohésion. Pour faire face à ce problème, nous avons prévu une étape de paramétrage au début du cycle d'évolution, permettant à l'expert d'accorder un poids à chaque critère d'évaluation. Ce poids reflète l'intérêt du critère relativement au domaine modélisé et à l'application utilisant l'ontologie.

Dans l'arborescence d'évaluation, la racine correspond au résultat final d'évaluation, les nœuds au-dessous de la racine correspondent aux aspects pris en considération lors de l'évaluation (structure, usage), les nœuds du niveau suivant correspondent aux critères d'évaluation et les feuilles aux mesures d'évaluation (nombre moyen de relations par concept, profondeur d'une hiérarchie,…etc.).

La première étape de l'évaluation consiste à calculer les métriques de chaque critère. Le résultat de l'évaluation de l'ontologie après l'application du changement, sera comparé à l'ancien résultat d'évaluation (avant l'application du changement). Si

la qualité est améliorée (ou préservée), le changement sera directement validé. Dans le cas contraire, c'est-à-dire lorsque le changement a un impact négatif sur la qualité, l'expert sera sollicité pour valider ou non le changement.

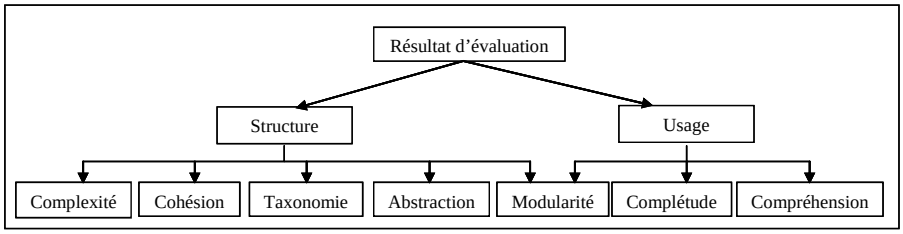

Figure 2. *Arborescence d'évaluation d'une ontologie.*

L'aspect *structure* permet d'évaluer l'ontologie en termes de complexité, cohésion, modularité, taxonomie et abstraction. L'aspect *usage* permet de mesurer le degré de compréhension, la complétude et la modularité (possibilité de réutilisation). La complexité mesure le degré de difficulté à comprendre la structure interne (les propriétés d'un concept) et externe (entre les concepts) de l'ontologie. La cohésion mesure le chevauchement entre les concepts de l'ontologie. Le chevauchement est défini par les liaisons entre concepts qui se font par le biais de relations. Un chevauchement important entre les concepts reflète une forte cohésion. La modularité mesure la possibilité de découper l'ontologie en parties (modules indépendants) pouvant être utilisées par d'autres ontologies. Un module est défini comme étant une sous hiérarchie relativement indépendante. La modularité facilite la maintenance et l'enrichissement de la structure de même que la réutilisation de l'ontologie. La taxonomie renseigne sur la richesse sémantique de l'ontologie. Elle est inversement proportionnelle à la part des relations autres que les hiérarchies *is-a*. L'abstraction mesure le niveau d'abstraction des concepts (le niveau de généralisation/spécialisation dans une hiérarchie). En ce qui concerne la complétude, une ontologie est dite complète si elle couvre les propriétés pertinentes du domaine d'intérêt. Ce critère est basé sur la conformité des noms des concepts avec les mots clés du domaine d'intérêt. La compréhension mesure la facilité de compréhension de l'ontologie à travers les différentes annotations et définitions des concepts.

Les métriques utilisées pour l'évaluation sont :

– **NRC :** Le nombre moyen de relations par concept.

– **NCC :** Le nombre moyen de chemins de la racine vers un concept.

– **NPC :** Le nombre moyen de propriétés par concept.

– **NCR :** Le nombre de concepts racines (nombre de hiérarchies).

– **NM :** Le nombre de modules disjoints formant l'ontologie.

– **H-ISA :** Le ratio entre le nombre de relations *is-a* et les relations sémantiques.

– **PMOY :** La profondeur moyenne d'une hiérarchie.

– **PREC :** La précision.

– **RAPP :** Le rappel.

– **CA :** Le pourcentage de concepts annotés.

– **RA :** Le pourcentage de relations annotées.

– **NTC :** Le nombre moyen de termes utilisés pour identifier un concept.

La complexité est mesurée avec NRC, NCC et NPC, la cohésion avec NRC et NCR, la modularité avec NM, la taxonomie avec H-ISA, l'abstraction avec PMOY, la complétude avec PREC et RAPP, et la compréhension avec CA, RA et NTC.

5. Prototype de validation basée sur l'évaluation

Pour la validation de l'approche proposée, un premier prototype a été implémenté permettant de valider la consistance d'ontologies OWL modifiées et d'évaluer leur qualité. La vérification des axiomes OWL et le calcul des différentes métriques d'évaluation ont été réalisés en utilisant l'API *Protégé-OWL* qui permet de naviguer dans l'ontologie. Cette API fournit un ensemble de méthodes permettant de manipuler et d'interroger des ontologies OWL.

L'application prend en input une ontologie OWL modifiée, le journal d'annotations des changements modélisé sous forme d'une base de données et les paramètres de l'expert relatifs aux poids des différents critères d'évaluation. Une première étape permet de vérifier que les axiomes OWL exprimés dans l'ontologie sont consistants en appliquant les algorithmes qui ont été défini et en les adaptant au modèle OWL. Une fois la consistance validée, l'évaluation de l'ontologie est effectuée en calculant les différentes métriques et en comparant la valeur de la qualité avec l'ancienne valeur contenue dans la base d'annotation.

6. Conclusion et travaux futurs

Dans cet article, nous proposons une approche d'évolution d'ontologie qui minimise l'intervention de l'expert dans la validation des changements à travers un processus d'évolution basé sur la vérification de la consistance et sur l'évaluation de la qualité de l'ontologie modifiée. La vérification de la consistance consiste à s'assurer que tous les axiomes restent valides après l'application du changement. L'évaluation permet de mesurer l'impact du changement sur la qualité de l'ontologie.

La consistance est centrée sur la validité des schémas d'axiomes décrivant les propriétés conceptuelles des entités ontologiques. L'évaluation de la qualité porte sur la structure et le contenu de l'ontologie décomposés en critères et métriques

d'évaluation. Au début du processus d'évolution, l'expert du domaine définit des pondérations pour chaque critère en lui octroyant un poids relatif à son importance par rapport au domaine et à l'usage de l'ontologie. Ainsi, le processus de d'évaluation permet de minimiser l'intervention de l'expert dans la validation des changements.

Un premier prototype de validation de changements pour des ontologies OWL a été implémenté. Il permet d'appliquer les algorithmes de vérification de schémas d'axiomes, d'évaluer les métriques définies dans le modèle d'évaluation et de mesurer l'impact du changement sur la qualité de l'ontologie en se basant sur l'ancienne valeur de qualité fournie par le journal d'annotation.

Dans des travaux futurs, nous envisageons d'enrichir la phase de validation de la consistance pour prendre en compte d'autres types d'axiomes et définir une correspondance entre les types de changement et les axiomes à vérifier. Par ailleurs, l'arborescence d'évaluation de la qualité d'ontologie peut être formalisé et étendue à d'autres critères de qualité.

7. Bibliographie

Akoka J., Comyn-Wattiau I., *A Knowledge-Based System for Auditing Computer and Management Information Systems*. Expert Systems with Applications, vol. 11, p. 361-375, 1996.

Bansiya J., Davis C.G., A Hierarchical Model for Object-Oriented Design Quality Assessment, *IEEE Transactions on Software Engineering*, vol. 28, n° 1, January 2002.

Brank J., Grobelnik M., Mladenic D., A Survey of Ontology Evaluation Techniques, *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, Ljubljana, Slovenia, 2005.

Brewster C., Alani H., Sasmahapatra S., Wilks Y., Data driven ontology evaluation, *Proceedings of Int. Conf. on Language Resources and Evaluation*, Lisbon, 2004.

Burton-Jones A., Storey V.C., Sugumaram V., Ahluwalia P., A semiotic metrics suite for assessing the quality of ontologies, *Data and Knowledge Engineering*, 2004.

Castano S., Dalakleidi K., Dasiopoulou S., Espinosa S., Ferrara A., Hess G.N., Karkaletsis V., Kaya A., Melzer S., Moller R., Montanelli S., Petasis G., Delivrable D4.1 Methodology and Architecture for Multimedia Ontology Evolution, 2006.

Flouris G., Plexousakis D., Handling Ontology Change: Survey and Proposal for a Future Research Direction, Technical Report FORTH-ICS/TR-362, 2005.

Fox M.S., Barbuceanu M., Gruninger M., Lin J., An organization ontology for enterprise modelling, Simulating organizations, *MIT Press*, 1998.

Fürst F., Trichet F., Reasoning on the Semantic Web needs to reason both on ontology-based assertions and on ontologies themselves, *Proceedings of the International Workshop 'Reasoning on the Wab' (RoW'06), co-located with the 15th International World Wide Web Conference (WWW'06)*, 2006.

Gangemi A., Catenacci C., Ciaramita M., Gil R., Lehmann J., Ontology evaluation: A review of methods and an integrated model for the quality diagnostic task, Technical Report available at http://www.loa-cnr.it/Publications.html, 2005.

Haase P., Sure Y., D3.1.1.b State of the Art on Ontology Evolution, SEKT Deliverable, 2004.

Haase P., Sure Y., Incremental Ontology Evolution – Evaluation, Institut AIFB, University of Karlsruhe, 2005.

Klein M., Noy N.F., Ontology Evolution: Not the Same as Schema Evolution, *Knowledge and Information Systems*, 2003.

Lozano-Tello A., Gomez-Perez A., Ontometric: A method to choose the appropriate ontology, *Journal of Database Management*, Vol. 15, N°2, p. 1-18, 2004.

Maedche A., Staab A., Measuring similarity between ontologies. *Proceedings of the European Conference on Knowledge Acquisition and Management – EKAW Madrid, Spain, LNCS/LNAI 2473*, Springer, *pp. 251-263.*, 2002.

Maedche A., Motik B., Stojanovic L., Managing multiple and distributed ontologies in the semantic web, *VLDB Journal*, p. 286-302, 2003.

Orme A.M., Yao H., Etzkorn L.H., Indicating Ontology Data Quality, Stability, and Completeness throughout Ontology Evolution, *Journal of Software Maintenance and Evolution: Research and Practice*, 2006.

Patel C., Supekar K., Lee Y., Park E.K., OntoKhoj: a semantic web portal for ontology searching, ranking and classification, *ACM Web Information. & Data Management.*, 2004.

Porzel R., Malaka R., A task-based approach for ontology evaluation, *Proceedings of ECAI 2004 Workshop on Ontology Learning and Population*, 2004.

Sure Y., Tempich C., State of the art in ontology engineering methodologies, SEKT informal deliverable 7.1.2, Institut AIFB, University of Karlsruhe, 2004.

Stojanovic L., Maedche A., Motik B., Stojanovic N., User-driven ontology evolution management, *In European Conference on Knowledge Engineering. and Management*, p. 285-300, 2002.

Stojanovic L., Maedche M, Stojanovic N, Studer R., Ontology evolution as reconfiguration-design problem solving, *In KCAP 2003*, ACM, p. 162-171, 2003.

Stojanovic L., Methods and Tools for Ontology Evolution, PhD thesis, University of Karlsruhe, 2004.

Supekar K., A peer-review approach for ontology evaluation, *Proceedings of the 8th International. Protégé Conference*, Madrid, Spain, July 18-21, 2005.

Xuan D. N., Bellatreche L. Pierra G., A versionning management model for ontology-based data warehouses, *Proceedings of* DaWak'06, 2006.

Yang Z., Zhan D., Ye C., Evaluation Metrics for Ontology Complexity and Evaluation Analysis, *IEEE International Conference on e-Business Engineering (ICEBE06)*, 2006.