Table des matières

Table des figures

Liste des Tableaux

Liste des Equations

Chapitre 1 Introduction

1. Introduction

Avec l'arrivée de l'Internet et du Web, le nombre de sources d'informations interconnectées ainsi que le nombre d'utilisateurs potentiels de ces sources a connu une augmentation exponentielle durant les dix dernières années. L'environnement informationnel actuel se caractérise par des données fortement distribuées. Ces données surabondantes sont généralement éparpillées, puisqu'il existe souvent de multiples systèmes conçus chacun pour être efficace pour les fonctions pour lesquelles il est spécialisé. Ces données se trouvent dans plusieurs domaines d'application tels que les entrepôts de données, l'intégration de données, le commerce électronique, le traitement de requêtes sémantiques, etc.

Le monde informatique regorge, ainsi, des données aux formats très hétérogènes, autrement dit utilisent des modèles différents pour la représentation de l'information, qu'il est nécessaire d'intégrer pour construire des applications. En effet, les données peuvent être de plusieurs types : **structurées** (données relationnelles, données objet), **semi-structurées** (HTML, XML, graphes) ou même **non structurées** (texte, images, son). Dans un tel contexte, le besoin d'intégration se fait de plus en plus sentir. Cependant, pour répondre à ce besoin, le développement des applications d'intégration (telles que pour un traitement élaboré de données, pour la construction des entrepôts de données ou des systèmes d'aide à la décision) se voit contraint de composer avec la répartition des sources et l'hétérogénéité de leurs structures et de gérer l'interopérabilité entre les données en différents formats qu'ils manipulent.

De nombreuses technologies ont permis de faire communiquer des applications relevant de systèmes d'informations différents permettant ainsi d'atteindre un niveau d'interopérabilité qu'il s'agit toutefois d'étendre et d'améliorer. En dépit de nombreux outils disponibles sur le marché, le problème de l'interopérabilité des applications demeure entier car il faut pérenniser l'existant pour l'intégration de standards mais aussi pour l'échange et la réutilisation.

Nous nous positionnons ici dans le cas d'intégration des données et nous cherchons à améliorer la transformation des schémas XML. Ces schémas XML sont une représentation logique enrichie par des méta connaissances sémantiques utilisées lors de la phase de Matching. Nous nous sommes donc intéressés au domaine du Matching et du Mapping des schémas XML pour lesquels il serait possible de réutiliser telles quelles les technologies déjà existantes. Ces deux processus qui se

suivent sont des pré requis à l'intégration et la transformation de documents XML.

2. Problématique

L'intégration des données se reporte à un problème combinatoire de données résidentes dans des sources autonomes et hétérogènes. Ce problème est rendu crucial avec la prolifération des sources de données sur Internet ou au sein des entreprises, le caractère hétérogène de ces données et le besoin de plus en plus pressant d'exploiter ces gisements de données pour des besoins décisionnels.

Notamment, l'appariement de schémas est l'un des problèmes majeurs rencontrés lors du processus d'intégration soit de données (par exemple, la médiation de données, les entrepôts de données, etc.), soit applications (par exemple, le e-commerce, le Web sémantique, etc.).

Tout ceci pose de sérieux problèmes aux utilisateurs qui cherchent à combiner, ou "intégrer" des informations provenant des sources différentes. Parmi eux on peut citer les problèmes liés à la gestion des schémas, l'évolution des schémas, le Mapping et le Matching des schémas.

Dans ce contexte, plusieurs approches ont été développées cernant ainsi les aspects de transformation de données qui résultent de l'hétérogénéité technique, syntaxique et sémantique des sources de données, de génération de Matching et de Mappings.

Pour simplifier et accélérer ces tâches d'intégration de données et garantir une interconnexion efficace entre tous les systèmes, il est nécessaire de bâtir des passerelles entre tous ces types de données. Ceci se fera en permettant de concevoir les schémas de Mapping de façon visuelle et en automatisant les transformations nous permettant de nous concentrer sur l'implémentation de la logique métier dans les applications.

3. Plan du rapport

Le présent rapport est organisé en trois grandes parties. La première est consacrée à l'état de l'art où est passée en revue la littérature relative aux technologies traitées. En effet, un état de l'art recensant les algorithmes de Matching de schémas est présenté dans le chapitre 2. Dans le chapitre 3, nous décrivons les modèles de découverte des expressions de mappage. Dans le chapitre 4, nous allons étudier les outils existants qui réalisent le Mapping et faire une comparaison entre eux.

Dans la deuxième partie, on va proposer une architecture répondant à toutes les problématiques posées. Dans le chapitre 5, nous allons détailler cette architecture et présenter ses spécificités.

Le chapitre 6 sera consacré à la conception de l'architecture proposée et le chapitre 7 à la réalisation et l'implémentation de cette architecture.

Enfin, la dernière partie (conclusion et perspectives) synthétise le travail accompli et les différents points que nous devons développer dans le futur.

4. Environnement du stage

Mon stage s'est déroulé au laboratoire LIRIS (Laboratoire d'Informatique en Images et Systèmes d'Information) à l' INSA (Institut National des Sciences Appliquées) de Lyon. Le LIRIS est né début 2003 à la suite du regroupement de plusieurs laboratoires de recherche lyonnais (LIGIM, LISI, RFV) et d'individualités du domaine des Sciences et Techniques de l'Information et de la Communication.

Il a deux thèmes principaux de recherche : l'image numérique et les systèmes d'information, qui sont déclinés suivant :

- Quatre axes scientifiques :

- Axe 1 - Données, Documents et Connaissances.
- Axe 2 - Images et vidéos : segmentation et extraction d'information.
- Axe 3 - Modélisation et réalité augmentée.
- Axe 4 - Systèmes d'information communicants.

- Deux actions transverses :

- Action A - Plate-forme d'Intégration d'outils logiciels pour le document numérique, en liaison avec l'*Institut des Sciences du Document Numérique* (ISDN).
- Action B - Plate-forme d'Intégration logicielle : dossier médical multimédia réparti, en liaison avec le thème fédérateur "*Ingénierie de la Santé*".

Chapitre 2 Etat de l'art sur le Matching

1. L'interopérabilité

L'interopérabilité de plusieurs sources de données hétérogènes et autonomes est un problème important dans plusieurs applications comme les systèmes de médiation, datawarehouse et les systèmes basés sur le web. Son but est de fournir une vue uniforme sur les données sources.

Les applications utilisant le système interopérable utilisent un schéma source qui représente ses données au monde extérieur et définissent le schéma cible qui représente leurs besoins. Il y a deux sortes de liens établis entre chaque schéma source et chaque schéma cible : le Matching (les correspondances sémantiques) et le Mapping.

2. Les schémas XML

Le schéma XML est une norme définie par le World Wide Web Consortium (W3C) [31] conçue comme une infrastructure de base pour la description du type et de la structure des documents XML.

Les schémas fournissent donc un modèle pour un document de données XML qui définit la mise en place des balises et du texte à l'intérieur de tous les documents faisant référence au schéma.

En termes d'utilisation, les schémas XML sont destinés à décrire la structure de données dans un format commun que les divers navigateurs Web, les applications et tous les clients utilisant XML peuvent reconnaître. Plus spécifiquement, les schémas définissent les règles qu' un document de données XML (notamment les noms d'éléments et les types de données) doit respecter. Ils définissent comment les éléments peuvent se combiner et quels attributs sont disponibles pour chaque élément.

L'utilisation d'un schéma est un atout pour l'interopérabilité. Il peut être fourni à d'autres applications, de façon qu'elles sachent structurer les données et donc les schémas, qu'elles transforment en retour.

Pour cette raison, les méthodes de Matching basées sur les schémas et donc, sur le nom des éléments, leur type, les méta données et des algorithmes sont développés. Ils permettent alors l'obtention d'un ensemble de règles d'association entre les éléments des schémas XML.

3. Les algorithmes de Matching des schémas

Le Matching des schémas est une technique qui effectue la découverte de correspondances sémantiques entre les éléments et les attributs des schémas. Le Matching est donc, une opération qui prend par exemple deux schémas de données en entrée et retourne à la fin les valeurs de similarités sémantiques entre les éléments des schémas.

Plusieurs travaux ont été réalisés afin de fournir des algorithmes de Matching gérant les correspondances ou incompatibilités des schémas.

Dans la littérature on distingue des catégories d'algorithmes de Matching:

3.1. Les approches basées sur les schémas de données

Dans cette catégorie, on trouvera les algorithmes travaillant sur les méta données (DTD, schémas XML, schémas de bases de données, …) spécifiques pour trouver l'indice de similarité le plus précis entre les éléments, Cupid [15] en est un exemple. Cet algorithme tente de trouver les correspondances sémantiques des éléments de différents schémas qui sont génériques. Les auteurs ont testé leur application sur les schémas XML et les schémas relationnels. Le processus de Matching se déroule en trois étapes: le calcul de la similarité, la similarité structurelle (les éléments atomiques, les éléments composés, les feuilles, les noeuds internes, les sous arbres), et le calcul du poids de similarité.

Similarity Flooding [16] [17] est un algorithme de Matching structurel pour schémas XML, SQL DDL, schemas RDF, UML et OEM. Il utilise l'idée d'influence de nœud sur ses adjacents.

Dans [18] XClust est une stratégie d'intégration basée sur le clustering des DTD ou bien sur les schémas. La similarité linguistique dans ce cas, est basée sur un thésaurus et la similarité structurelle est basée sur (les éléments atomiques, les éléments composés, sur les sous arbres et nombre de feuilles).

3.2. Les approches basées sur les instances ou ontologies

Cette autre catégorie d'algorithmes travaille plutôt sur les instances[19][20] ou bien encore sur les ontologies [21][22]. Par ailleurs, il existe également des frameworks comme COMA[6] ou COMA++ [23] qui utilisent plusieurs algorithmes avec des techniques différentes. Les schémas sont

traduits dans des graphes acycliques traités simultanément par ces algorithmes. Les résultats intermédiaires peuvent être utilisés avec des possibilités de sélection et agrégation. L'utilisateur d'un tel outil reste néanmoins un exercice difficile pour un utilisateur non averti.

4. EX-SMAL [7] (EDI/XML semi-automatic Schema Matching Algorithm)

Le Matching dans EXSMAL est un processus qui permet la découverte des relations sémantiques entre deux schémas XML, il prend deux schémas de données en entrée et retourne à la fin des valeurs de similarités sémantiques entre les éléments des schémas en entrée.

L 'algorithme de Matching EX-SMAL (EDI/XML semi-automatic Schema Matching Algorithm) permet de découvrir, donc, semi-automatiquement les correspondances entre les messages EDI (Echange de données informatisées) basées sur les schémas XML**.** La compatibilité entre deux messages de deux standards différents repose sur la sémantique de ces deux messages.

L'algorithme génère des relations de Matching entre les noeuds des schémas XML dont la cardinalité varie entre 1-1, 1-n, n-m.

Il traite la structure de chaque élément à faire correspondre afin de raffiner l'efficacité de résultat. Il se base sur la similarité de base entre les éléments individuels, en s'appuyant sur les descriptions textuelles et les types de données des éléments des guides d'utilisations, spécifié à l'aide de schémas XML, et la similarité de structure des éléments en comparant les voisinages structurels des éléments de messages en entrée. Les deux similarités sont utilisées pour calculer la similarité finale entre chaque paire d'éléments et cette dernière est enfin filtrée afin d'obtenir le résultat final de Matching.

Etant donné un schéma XML considéré comme structure de données interne, l'algorithme travaillera avec une structure arborescente. L'algorithme se déroule en trois étapes qui sont décrites dans la figure 1. Il consiste à calculer la similarité entre tous les éléments de deux schémas, pour chaque paire d'éléments, on calculera d'abord la similarité de base qui tient compte des données qu'un élément de schéma possède individuellement soit la description textuelle et le type de données. Ensuite, la similarité de structure, qui se base sur le calcul de leurs contextes et le calcul de similarité de base de chaque élément dans les deux schémas. Puis, utiliser les valeurs de ces deux similarités pour calculer la valeur de similarité finale entre ces paires.

Algorithme :

Entrée : S1, S2 : deux schémas XML
Sortie : Ensemble de triplets < $E1_i$, $E2_j$, V_{sim}>
Avec $E1_i$: élément de schéma S1
$E2_j$: élément de schéma S2
V_{sim} : la valeur de similarité entre $E1_i$ et $E2_j$ ∈ [0, 1]

Matching (S1, S2) {
- Calculer la similarité de base entre $E1_i$ ∈ S1 et $E2_j$ ∈ S2
- Calculer la similarité structurelle entre $E1_i$ ∈ S1 et $E2_j$ ∈ S2
- Pour chaque paire <$E1_i$, $E2_j$> calculer V_{sim} en fonction de leurs similarité de base et similarité structurelle trouvées
- Sélectionner les paires de correspondance <$E1_i$, $E2_j$> qui sont les plus plausibles

}

Figure 1 Description brève de l'algorithme

5. Conclusion

Dans ce chapitre, nous avons défini différentes catégories d'algorithmes de Matching. Le but de ces algorithmes est la mesure de similarité linguistique et structurelle entre les éléments, cette similarité est exprimée par un coefficient entre 0 et 1. Quant à la transformation des documents XML, nous avons besoin des opérations de transformations qui ne se basent pas sur les valeurs numériques mais sur les relations sémantiques entre les entités de schémas. Nous décrivons, dans le chapitre qui suit, les différents modèles d'expressions de mappage.

Chapitre 3 Etat de l'art sur les modèles de mappage

1. Mapping des schémas

Si le Matching consiste à trouver les relations entre des entités avec un certain coefficient qui indique qu'elles sont plus ou moins similaires, le Mapping consiste à trouver la vraie relation sémantique qui permettra le passage de l'un à l'autre.

Les Mappings sont des expressions décrivant le moyen dont les instances du schéma cible sont dérivées à partir des instances des sources. Les Mappings sont définis en utilisant les correspondances existantes entre les schémas. Quand les Mappings sont définis entre le schéma cible et le schéma source, cela consiste essentiellement dans la restructuration des données d'une présentation à une autre. La définition de ces Mappings est aussi connue comme l'échange des données, la transformation des données et la migration des données. Le Mapping peut être plus complexe: le Mapping ne transforme pas seulement la donnée source d'une structure à une autre mais peut également combiner différentes sources. Plusieurs problèmes sont à soulever:

- Trouver tous les éléments sources correspondants à l'élément cible,
- Combiner les instances des éléments source pour former les instances du schéma cible,
- Satisfaire la structure du schéma et les contraintes de cardinalité du schéma cible.

Résoudre ces problèmes requiert une compréhension profonde non seulement de la sémantique des schémas sources mais aussi des liens sémantiques entre les schémas sources et cibles. La complexité de ce processus augmente quand le nombre de données sources est élevé. Si les schémas sources et cibles sont en format XML, la définition du Mapping devient plus complexe à cause de la nature hiérarchique des données [9].

Ceci revient à des problèmes de cardinalité de Mapping qui peuvent être classés en deux catégories. La première catégorie est le Mapping direct qui correspond à une cardinalité de Mapping (1:1). Ce type de cardinalité est le plus étudié dans les approches existantes à cause de la difficulté pour déterminer automatiquement les expressions de mappage. La seconde catégorie est le Mapping indirect qui couvre les cardinalités de type (1:n), (n:1) et (n:m) qui signifie qu'il peut y avoir n

éléments d'un schéma et m éléments d'un autre schéma qui correspondent [10].

De là vient le besoin de définir un processus de découverte d'expressions de mappage après le Matching consistant à trouver une expression (logique, mathématique, opération sur les chaînes de caractères, etc) permettant d'associer un ensemble d'éléments du schéma source pour obtenir un élément du schéma cible (ex: name=FirstName CONCAT lastName), pour la découverte d'expressions de mappage [2]. Les expressions de mappage sont requises pour exprimer comment les éléments matchés (mis en correspondance) peuvent être mappés (transformés).

Plusieurs modèles ont été proposés pour exprimer les expressions de mappage offrant des opérations de transformations qui sont données par les relations sémantiques telles *que (équivalent, plus général, incompatible, compose, est composé de,…).*

2. Modèles de découverte des expressions de mappage

2.1 Value Correspondences

L'approche présentée dans [11] crée un Mapping interactif basé sur les valeurs de correspondances qui montrent comment la valeur d'un attribut cible peut être créée à partir d'un ensemble de valeurs d'attributs sources. Les expressions de mappage sont étudiées comme des valeurs de correspondances.

La valeur de correspondances est une paire qui consiste en:

- Une fonction définissant comment une valeur (ou une combinaison de valeurs) de la base de données source peut être utilisée pour former une valeur dans la cible.
- Un filtre indiquant quelles valeurs sources doivent être utilisées.

Les valeurs de correspondances peuvent être entrées par un utilisateur ou peuvent être suggérées utilisant des techniques linguistiques appliquées aux données et méta-données comme les noms des composants de schémas.

Le modèle d'expression de mappage dans [11] est défini comme suit:

$$v_i = \langle f_i, p_i \rangle$$

Équation 1 Modèle d'expression de Mappage dans Miller et al

- Fi: la fonction de correspondance dénotant la valeur de substitution
- *Pi : un filtre*

L'implémentation de cette approche est un outil semi-automatique Clio pour la création de schémas de Mappings. Il emploie un Mapping qui dépend de l'utilisation des valeurs de correspondances décrivant quelle valeur de l'attribut cible peut être créée par un ensemble de valeurs d'attributs source. Clio utilise un raisonnement autour des requêtes à créer, manager et classer les Mappings possibles. Cependant, le choix final des Mappings revient à l'utilisateur qui doit comprendre la sémantique de l'application cible [12].
L'inconvénient de ce prototype est qu'il ne résout pas le Mapping conditionnel, le Mapping avec rupture de clés et le Mapping simple.

2.2 Le Modèle d'expression de Mappage [1]

Un modèle d'expressions de mappage a été proposé dans [1]. Ce modèle est une extension de l'approche présentée dans le paragraphe précédent du fait qu'il rajoute des conditions et effectue un filtrage par rapport à la cible. Cette approche sert à définir les expressions de mappage entre les éléments similaires. L'idée des valeurs de correspondances de Clio est similaire aux expressions mappage. Il définit dans les méta-données les mappages entre les éléments de schéma source et schéma cible.

$$A = \text{Concat}\,(\ldots, \underbrace{[f_i(a_r^{Sr}, a_p^{Sp}, \ldots, a_e^{Se}), l_i, c_i]}_{\alpha_i(A)}, \ldots)$$

Équation 2 Modèle d'expression de Mappage

Il couvre, par ailleurs, différents types d'expression de mappage :

- **Décomposition/concaténation:** la valeur du champ est établie par décomposition de la valeur de la source et par sa concaténation avec d'autres valeurs.
- **Opérations arithmétiques avec des données multiples:** Dans ce cas, une fonction arithmétique est définie. Cette fonction calcule la valeur du résultat du champ cible.
- **Mapping conditionnel:** Parfois la valeur du champ cible dépend de la valeur d'un autre champ.

- **Mapping avec décomposition de la clé:** La représentation sur rang dépend de la valeur

du champ.

- **Mapping utilisant une table de correspondances avec des colonnes multiples:** La valeur du champ cible est une fonction utilisant plusieurs valeurs dans la table de conversion.

Il permet aussi d'exprimer sans difficulté les quatre catégories de traduction : La traduction simple (1:1), la traduction avec découpage (1:n), la traduction avec groupement (n:1) et la traduction avec découpage et groupement (n:m). Ces traductions ont été utilisées dans le cas de messages de type EDI.

Ce modèle couvre, donc, différents types d'expressions de mappages, les différents cas de traduction, Il est applicable sur un Mapping entre les entités et non pas entre les schémas et permet de couvrir des cas utiles de génération d'éléments cibles à partir de fonctions de Mapping.

2.3 Le Modèle de données LIMXS

Le modèle LIMXS [8] (Layered Interoperability Model for XML Schemas) offre une vue sémantique et une vue logique des schémas XML basée sur la modélisation conceptuelle.

- La vue sémantique, décrite en terme de concepts et relations sémantiques, donne une définition explicite et formelle des éléments des schémas sémantiques et significations qui nous permettent de mapper selon le sens des éléments des schémas et non pas seulement leurs labels.

Deux phases sont adaptées:

- L'analyse linguistique qui correspond au niveau concept : Elle définit des relations sémantiques entre les concepts. Cinq types de relations sémantiques sont considérés: équivalent, général, spécifique, compatible et incompatible.
- L'analyse structurelle qui correspond au niveau structure: Le but est d'étendre les relations sémantiques découvertes par les analyses linguistiques par l'examen de l'organisation des données fournies par les vues sémantiques. L'organisation des données est décrite à travers les relations d'agrégation (composition de relations entre deux concepts), association (relation sémantique entre deux concepts qui sont conceptuellement au même niveau), généralisation (relation entre un concept général C1 et un concept plus spécifique C2) et contraintes (ensemble de contraintes sur les concepts et les relations).

- La vue logique décrit les schémas construits indépendamment des détails syntaxiques et nous permet de représenter le Matching à un niveau logique.

Ce modèle se base donc sur un ensemble d'opérations de transformations sur les entités des schémas qui peuvent être composés ensemble pour représenter plus de transformations (les opérations conceptuelles primitives, opérations logiques).

- **Les opérations conceptuelles primitives:**
  - **Add:** ajoute une entité au schéma cible, l'entité peut être une relation, un concept, une propriété.
  - **Merge:** fusion de deux entités en une seule entité: concaténation.
  - **Split:** c'est l'opération inverse de Merge.
  - **Rename:** change le nom d'un concept ou d'une propriété
  - **Connect:** il s'agit d'un Mapping 1-1 qui relie deux entités équivalentes sans aucune transformation
- . **Les opérations logiques:**

Les opérations logiques traitent les différences dans les contraintes des schémas comme la cardinalité, les valeurs nulles ou par défaut mais peuvent être une information supplémentaire que seul l'utilisateur peut fournir.

```
< Mapping ID = "map1 ">
<SourceSchema source = "S1.xsd"/>
<TargetSchema target = "S3.xsd"/>
<HasMappings OneToOneMapping ID = "map11"/>
</Mapping>

< OneToOneMapping ID = "map11" >
<ConceptMapping>
<SourceConcept concept = "Staff"/>
<TargetConcept concept = "Employee"/>
<Transformation>
<HasMappings OneTomany Mapping ID = "map111"/>
<Operation name ="rename"/>
</Transformation>
</ConceptMapping>
</OneToOneMapping>
```

```
<ManyToOneMapping ID = "map111" >
<AttributeMapping>
<SourceAttribute Attribute = "FirstName"/>
<SourceAttribute Attribute = "LastName"/>
<TargetAttribute Attribute = "Name"/>
<Transformation>
<Operation name ="merge"/>
</Transformation>
</AttributeMapping>
</ ManyToOneMapping >
```

Tableau 1 Exemple de représentation de LIMXS en XML

Avantages: Ce modèle est capable de: Modéliser tous les facteurs de schémas XML tels que la spécialisation/généralisation des relations, espaces de noms,etc...

- Offrir les deux vues sémantique et logique des schémas XML
- Réaliser le Mapping entre les entités des schémas.

Inconvénients:

Ce modèle traite deux types de cardinalités correspondants au

- *Matching simple*: correspond aux relations de Matching 1-1 entre un noeud source et un noeud cible. On peut utiliser des opérations simples comme **Connect** dans ces cas.
- *Matching complexe:* correspond aux relations de Matching 1-n et n-1 entre un noeud source et un noeud cible. Pour ce type de Matching, on utilise les opérations de transformation comme **Merge et Split**.

Mais il ne traite pas le cas de la cardinalité n-m.

2.4 Le modèle XHS (Intégration via l'HYPERSCHEMA XML)

L'hyperschéma XML (noté XHS) est constitué de l'ensemble des schémas des BDs sous forme XSD étendu et de l'ensemble des opérateurs de transformation d'un schéma à un autre sous forme XSL. L'approche proposée dans [24] permet de fournir un modèle d'intégration capable de fédérer les différentes sources de données hétérogènes représentant des dimensions structurelle et sémantique des schémas sources. L'approche s'appuie sur un modèle semi structuré (XML) qui permet une intégration aisée de modèles variés ainsi que sur deux parties distinctes à savoir :

- Le module d'extraction de schémas sources qui homogénéise la représentation des différents schémas sources à intégrer en l'exprimant sous forme d'une définition de schémas XML.
- Le module d'intégration permettant de construire l'hyperschéma qui constitue l'ensemble des schémas XML étendus (EXSi) et l'ensemble des schémas XSL stipulant des règles de transformation entre les entités des EXS. Le schéma XML étendu est composé de trois types d'entités:

- ***Concept(c):*** engendre des propriétés et/ou d'autres concepts imbriqué, et porte des relations.

- ***Relation(r):*** correspond aux associations structurelles et/ou sémantiques existantes entre

deux ou plusieurs concepts.

- ***Propriété (p):*** peut être incluse dans des concepts ou relations.

Le schéma EXS est donc défini par le triplet <concepts, relations, propriétés> illustré dans la figure2 :

- c_i est le concept nommé C de type <Cconcept>
- $r(c_i....c_j)$ est la relation existante entre deux ou plusieurs concepts $c_i....c_j$ nommé « $c_i....c_j$RelationShip » qui possède une catégorie prédéfinie et inclut les concepts en question.
- P_k^i est la $k^{ème}$ propriété nommée p du concept c_i, elle possède un domaine de définition et des contraintes (valeur par défaut, obligatoire,etc).

```
<!--définition du concept personne -->
<xsd:complexType name="personneConcept"
                        level="1"  complete="false">
  <xsd:sequence>
     <xsd:element name="name" type="xsd:string"
                     minOccurs="1" maxOccurs="1"/>
     <xsd:element name="profession" type=" xsd:string"
                     minOccurs="1" maxOccurs="1"/>
     <xsd:element name="address"  type="addressConcept"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:positiveInteger"
                    use="required"/>
</xsd:complexType>

<!--définition du concept adresse -->
<xsd:complexType name="adresseConcept"
                        level="2" complete="true">
...
</xsd:complexType>

<!--définition de la relation r(personne, adresse) -->
<xsd:complexType name="personneadresseRelationShip"
                        type="dependance">
 ...
   <xsd:element name="personne" type="personneConcept"
                    source="true" target="false" level="1"/>
 ...
```

Figure 2 EXS représentant le concept personne

A l'issue de cette étape, six opérateurs de base seront appliqués d'un schéma EXS à un autre lors des appels aux primitives de transformations qui correspondent aux morphismes entre deux entités du même domaine afin de réaliser le Mapping entre les différents éléments:

- **Addition (add)**: permet d'ajouter une entité (concept, propriété d'un concept ou relation existante entre concepts) au schéma source.

- **Suppression (delete)**: permet de supprimer une ou plusieurs entités lors du processus de mises en correspondance (Matching) entre le schéma source et cible (*(EXSsource → EXScible)*. La suppression est l'opérateur inverse de l'ajout.
- **Fusion (merge)**: permet de fusionner deux entités distinctes en une entité atomique afin de regrouper des propriétés appartenant au schéma source en un concept dans le schéma cible.
- **Eclatement (split)**: permet la décomposition d'une entité du schéma source en un ensemble d'entités équivalentes dans le schéma cible. C'est l'opérateur inverse de merge.

2.5 Model Management System

Dans cette approche, les opérateurs sont génériques et sont appliqués aux schémas (nommés modèles) et non aux éléments des schémas.

Selon [13], l'implémentation des applications revient à traduire les modèles de données en des représentations orientées objets et manipuler les modèles et les Mappings dans ces représentations. La manipulation inclut la conception du Mapping entre les modèles, la génération d'un modèle à partir d'un autre à l'aide d'un Mapping entre eux, la modification du modèle ou Mapping, l'interprétation du Mapping et la génération du code du Mapping.

L'implémentation des ces abstractions et opérateurs s'appelle MMS:**Model Management System** dans lequel les modèles et Mappings sont des structures syntaxiques mais n' ayant pas de sémantique basée sur les contraintes de langages ou langages de requêtes.

Les principaux opérateurs du système de gestion de modèles MMS:

- **Match:** prend deux modèles comme entrée et retourne un Mapping entre eux.
- **Compose:** saisit un Mapping entre les modèles A et B et un Mapping entre les modèles B et C et retourne un Mapping entre A et C.
- **Diff:** saisit un modèle A et un Mapping entre A et un certain modèle B et retourne un sous-modèle de A qui ne participe pas au Mapping.
- **ModelGen:** saisit le modèle A et retourne un nouveau modèle B basé sur A et Mapping entre A et B.
- **Merge:** saisit deux modèles A et B et un Mapping entre eux et retourne une union C

de A et B avec Mapping entre C et A et C et B.

L'expression suivante est un exemple de formalisation de ce modèle:

$$v.\mathrm{M} = \pi_{\mathrm{C,N}}(\sigma_{\mathrm{V}=\text{“ACME”}}(s.\mathrm{O} \bowtie s.\mathrm{S}))$$

Équation 3 Formalisation du modèle

Inconvénients:

Cette approche est basée sur un Mapping entre les modèles (ou schémas) et non sur les éléments.

2.6 TUPELO

TUPELO [26] est un système de Mapping de données, qui permet la découverte des expressions de mappage complexes. Ce système utilise un langage de transformation nommé L qui inclut des opérateurs pour surmonter l'hétérogénéité structurelle entre les données sources relationnelles. Le système prend en entrée les schémas source/cible et les instances source/cible référencées comme étant des instances critiques. Ces instances critiques illustrent la même information sur les deux schémas et sont utilisées pour guider la découverte des Mappings de données. La découverte des expressions de mappage est traitée comme une séquence d'acheminement d'exploration de l'espace de transformation des opérateurs dans le langage L sur l'instance source. Dans l'implémentation du système, ils utilisent des variations d'heuristiques pour découvrir la séquence de transformations qui mappent la source à la cible. L'expression de mappage découverte est retournée par le système.

Le problème de Mapping des données est défini comme suit:

Etant donnée une source ***S*** et un schéma cible ***T*** et un langage de transformation approprié ***L*** , trouver un Mapping ***i∈L*** tel que pour chaque instance ***s*** de ***S*** et une instance correspondante ***t*** de ***T, i(s)=t.***

La nouveauté de ce système est la génération de Mapping pour des Matchings de schémas simples, il ne génère cependant pas les Mappings de données qui incluent les transformations sémantiques complexes.

Le modèle est exprimé dans un modèle mathématique:

$$\lambda^{B}_{f,\bar{A}}(R).$$

Équation 4 Modèle mathématique

- λ: un opérateur
- A= <A_1....A_n> : un attribut
- B: un attribut
- R: relation
- f: fonction de Mapping

3. Synthèse des expressions de mappage

Dans ce chapitre, nous nous sommes particulièrement intéressés au Mapping des schémas. Nous avons vu que Clio par exemple est présenté comme un outil semi automatique de Mapping où les correspondances sont supposées déjà identifiées et exprimées dans un langage de logique de premier ordre. Ce que l'on note c'est que les plateformes de Matching ne sont pas forcément complétées pour le Mapping et avec des études de cas pour des processus d'interopérabilité. Nous allons donc étudier, dans le chapitre suivant, quelques outils et plateformes de Mappings, discuter leurs avantages, leurs lacunes et effectuer un comparatif de ces outils.

Chapitre 4 Etat de l'art sur les outils de Mapping

1. Introduction

Il existe sur le marché des plateformes de Mapping, des plateformes pour le Matching issues de travaux de recherche et peu de plateformes incluant les deux processus. Nous allons dans ce qui suit, donner un aperçu de la première catégorie de plateformes.

2. Les plateformes et Outils de Mapping

2.1 Altova MapForce

MapForce est un produit de Altova mais aussi un outil de Mapping visuel puissant, travaillant avec des fichiers XML, des bases de données, des fichiers plats, des données EDI, permettant de développer des outils personnalisés d'intégration de données. Il permet, ainsi, de bâtir des passerelles entre différents types de données afin d'assurer leur échange sous des formats hétérogènes pour fournir une interconnexion efficace entre tous les systèmes

MapForce se distingue par le fait qu'il offre la première implémentation de XSLT 2.0 et XPath 2.0. Il est aussi le premier outil du marché (et le seul) qui permet de générer le code de Mapping sous divers formats XSLT 1.0, XSLT 2.0, XQuery, Java, C#, et C++ à partir d'une interface unique.

MapForce permet, en plus, de simplifier et d'accélérer les tâches d'intégration de données grâce à une conception des schémas de Mapping de façon visuelle et en automatisant les transformations. En utilisant MapForce, on définit les Mappings entre des modèles de contenu de façon visuelle juste par drag and drop entre les éléments correspondants entre la source et la cible sur une interface conviviale [27] (figure 3).

Il permet, aussi, de "mixer" plusieurs sources et plusieurs cibles pour permettre le Mapping à partir de toutes les combinaisons de format possibles.

De plus, MapForce étend la notion du Mapping au-delà de la gestion de 2 fichiers XML en offrant la possibilité de gérer plusieurs fichiers qui ainsi connectés jouent parfois le rôle d'input et d'output à la fois.

On peut aussi ajouter des règles de traitement des données. MapForce génère automatiquement le code logiciel nécessaire pour mettre en place la transformation des données sources vers les données cibles. Le moteur MapForce intégré permet de pré visualiser le résultat de tout Mapping en un seul clic, ce qui simplifie la conception et les tests de façon conséquente.

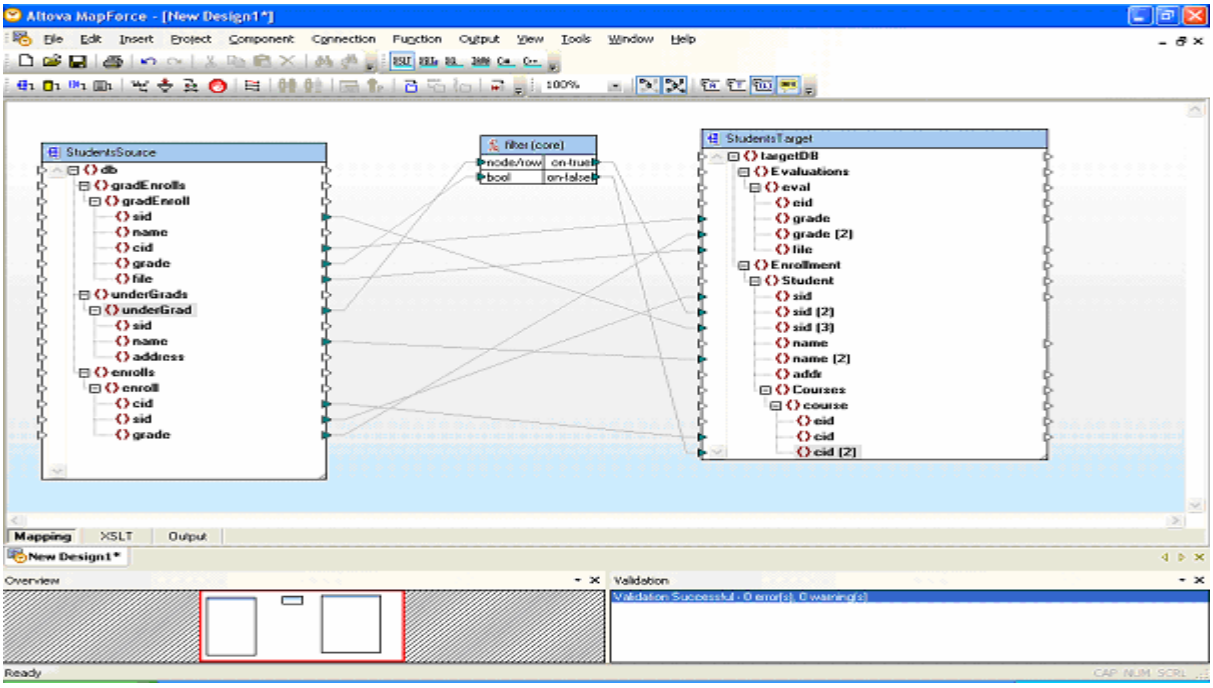


Figure 3 Mapping dans MapForce

Mapforce supporte les fonctionnalités suivantes:

- Construire des services web à travers une surface graphique
- Faire du Mapping XML, base de données, fichiers et les sources EDI aux opérations des services Web.
- Faire du Mapping pour toute combinaison de XML, base de données, fichiers et des données EDI à travers l'interface graphique.
- Générer automatiquement XSLT 1.0, XSLT 2.0, XQuery, Java, C++ ou C# pour implémenter les données de Mapping (XSLT et XQuery pour le Mapping XML-to-XML).

- Exécuter, observer et sauvegarder les outputs des transformations des données de Mapping.
- Traitement et filtrage des données à l'aide une riche bibliothèque de fonctions.

Limites :

- MapForce ne permet pas d'éditer des schémas directement. On est obligé d'utiliser différents éditeurs de schémas afin de rapporter des modifications aux schémas puis rouvrir le schéma dans MapForce pour continuer le Mapping.
- Duplication du nœud cible lors d'un Mapping de plusieurs nœuds vers un seul.

2.2 Schema Mapper

SchemaMapper est un nouvel outil, un prototype intelligent de Mapping de schémas. Sa première motivation provient de ETANA-DL qui est une librairie digitale pour promouvoir l'intégration de l'information et des services de divers sites archéologiques. C'est le premier outil qui offre une représentation visuelle de schémas de Mapping en arbres hyperboliques pour donner une meilleure vue afin de percevoir plus de détails sur l'écran. Ce qui est utile surtout pour les schémas à grande échelle en permettant une navigation plus facile et moins de confusions [5].

Son architecture est divisée en quatre composants:

- **Composant de Visualisation :** Responsable de toutes les fonctions reliées à la visualisation du processus entier de Mapping.
- **Composant de Recommandation :** Responsable de donner des recommandations données à l'utilisateur pour indiquer le Matching possible pour un nœud local spécial.
- **Composant de Mapping :** Responsable de faire des Mappings. Dés l'établissement du Mapping entre deux nœuds, ce composant sauvegarde les informations dans les structures de données.
- **Composant de génération XML :** Dès que le Mapping est réalisé, ce composant crée une feuille de style XSLT sur le Mapping courant déjà stocké dans le composant de Mapping.

Pour réaliser le Mapping de nœuds (figure 4), il faut sélectionner le schéma local du nœud à mapper et faire un simple clic droit sur le schéma global du nœud vers lequel on veut réaliser le Mapping. Une fois mappé, la couleur des nœuds mappés change en pourpre et le Mapping sera enregistré

dans la table de Mapping (qui se situe dans le coin gauche de l'interface graphique). Le processus de Mapping dans SchemaMapper est soumis à un ensemble de règles telles que seules les feuilles de nœuds peuvent être mappées directement vers d'autres mais aussi la feuille du nœud non locale ne peut pas être mappée à la feuille du nœud global.

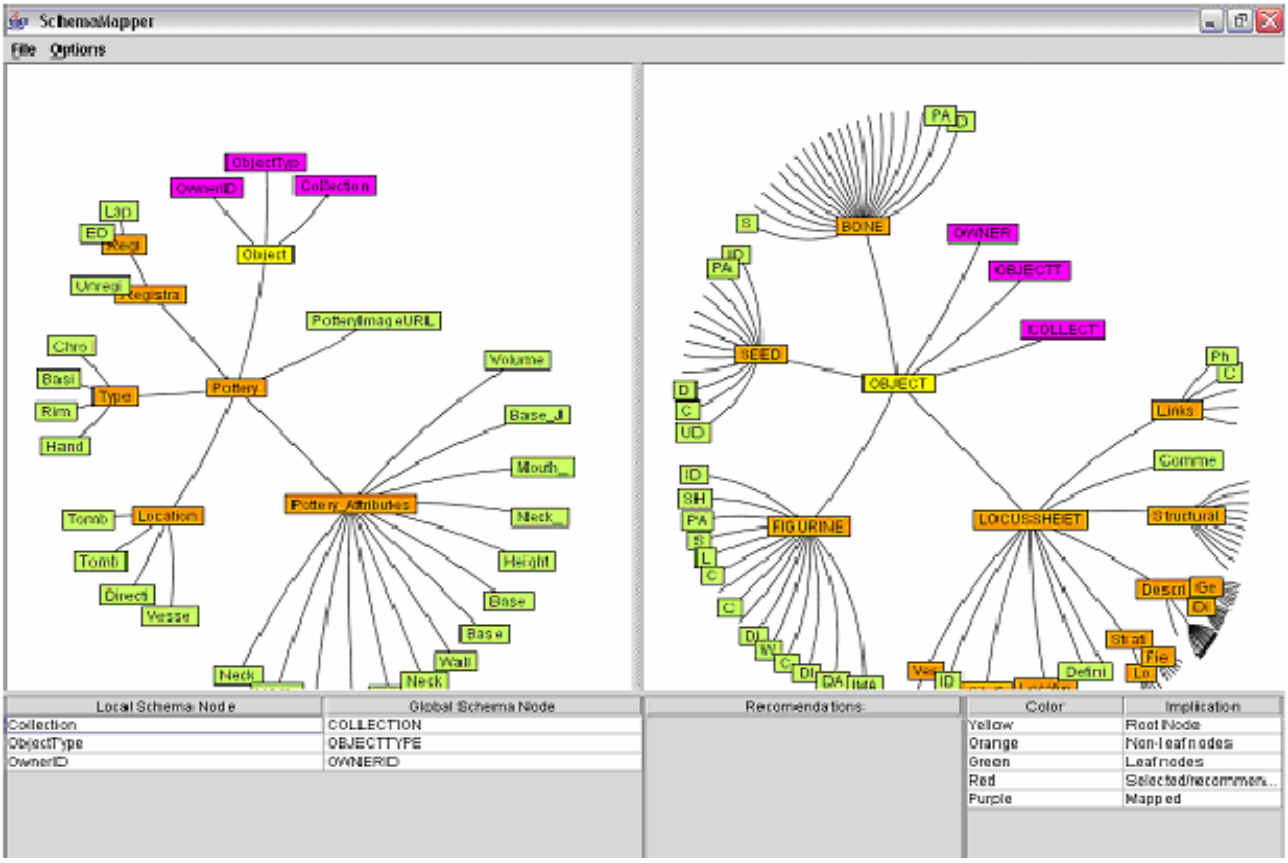


Figure 4 Mapping des noeuds

Cet outil a pour avantage d'éviter l'approche de line-drawing pour indiquer le Mapping. Il offre de ce fait une architecture flexible qui permet une extensibilité pour des fonctions supplémentaires. Il utilise aussi la notion de coloration pour distinguer entres les différents types des nœuds des schémas (root, leaf, non-leaf, selected, recommended, et mapped) et permet de renommer, supprimer un nœud et ajouter un schéma local sub-tree dans le schéma global. Il permet d'éditer les schémas dans le même outil.

Limites:

- Cet outil se limite à un Mapping de schémas XML à XML. En effet, il ne supporte pas des Mappings de schémas de base de données de même que le Mapping des schémas de méta données.

- Il manque d'options de génération de code.
- Il ne permet pas les Mappings complexes qui nécessitent des opérateurs mathématiques.
- La vue des arbres hyperboliques rend difficile la vue de toutes les recommandations en même temps.

2.3 Stylus Studio

C'est un produit commercial qui inclut un outil de Mapping visuel **XML Schema Mapping** qui permet le Mapping visuel entre schéma source et schéma cible par un drag et drop et offre une implémentation facile des données XML impliquant de multiples sources de données et personnalise les processus de données utilisant XSLT ou le code XQuery.

Il réalise le Mapping entre schémas XML mais aussi d'autres formats de données comme les DTD (Document Type Definition), les instances de documents XML, les bases de données relationnelles avec XML, les services web dans une interface graphique conviviale composée de deux parties : une pour visualiser le Mapping et une autre pour visualiser le code source généré en XSLT ou XQuery conforme au Mapping et qui peut être édité (Figure 5).

Il offre une vitesse de traitement des données rapide en Input et en Output et intègre un environnement de développement XML. Il comprend entre autre un éditeur et un débogueur XSLT, un outil de conception graphique de schémas XML, un composeur d'appel pour services web (Web Services Call Composer) et un éditeur, débogueur et mapper XQuery [28].

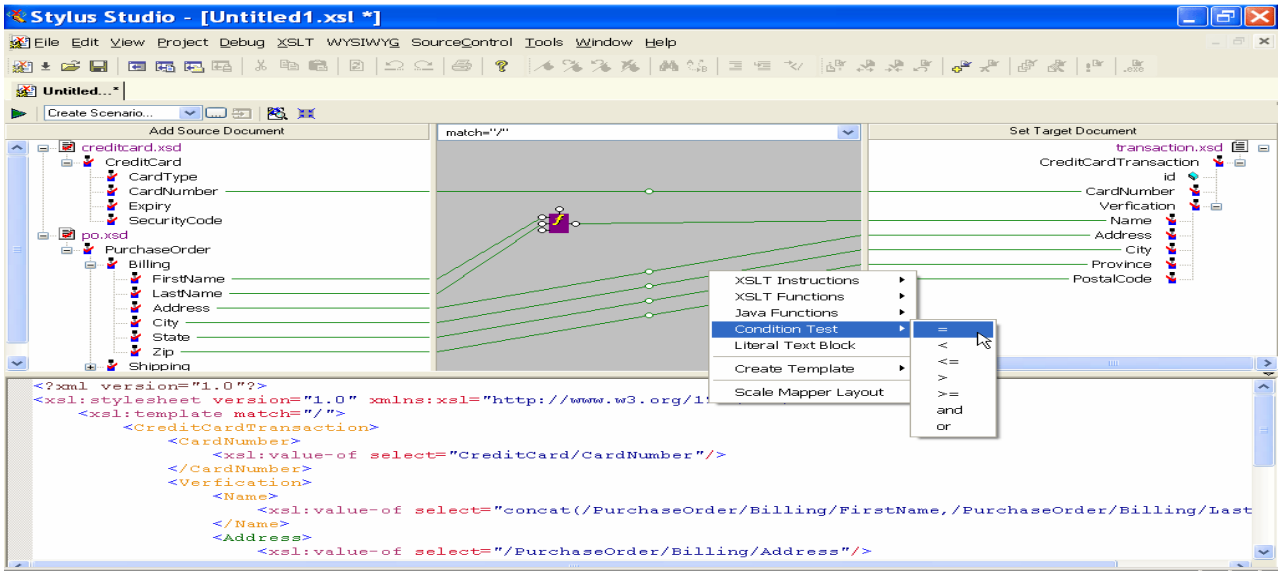


Figure 5 Mapping dans Stylus Studio

Limites:

- Il ne permet pas de mapper plusieurs schémas.
- Il génère uniquement le code XSLT et XQuery.

2.4 Visual XSLT

C'est un outil couplé avec Visual Studio .Net [32], ce qui lui procure une vaste palette de fonctions lui permettant notamment de réaliser des Mappings avec la technologie des templates XSLT.

Il permet un Mapping visuel des schémas et transforme des documents XML par une interface drag et drop. Le Mapping est réalisé à partir d'un schéma source vers un schéma cible en cliquant sur l'élément ou l'attribut de la source et en tirant (dragging) vers l'élément ou l'attribut de la cible (figure 6). Ce lien est indiqué par l'icône « mapper » dans fenêtre du schéma cible dans l'interface graphique. Un nœud dans le schéma source peut être mappé vers plusieurs nœuds du schéma cible et vice vers ça. Il fournit un traitement des informations rapide, offre une facilité d'accès et est très maniable.

Il offre en plus un debugging JIT (Just In Time) en testant le code XSLT placé dans d'autres applications .Net ou d'autres librairies sans nécessiter d'une technologie spéciale ou d'un accès au code source.

On peut aussi utiliser des sources de données comme inputs ou outputs telles que des documents instance XML, des schémas XML ou DTD, des services Web afin de faire du Mapping de données dynamiques générées de services web et des Bases de données relationnelles.

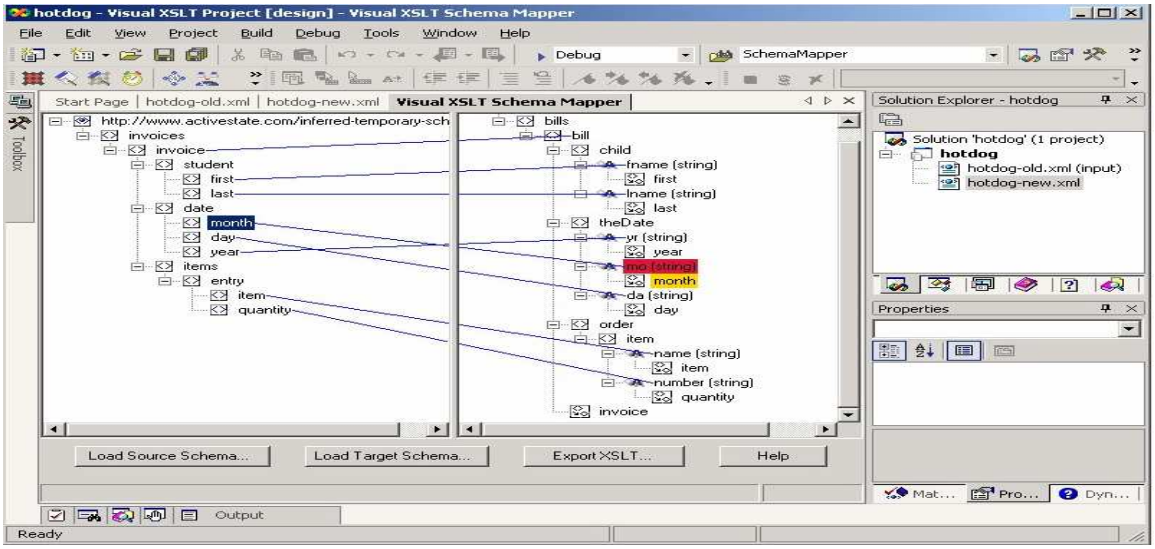


Figure 6 Mapping dans Visual XSLT

Limites:

- On ne peut mapper que vers un seul schéma cible.
- Ne permet pas la génération du code XQuery, java et autre

2.5 TIBCO XMLTransform

TIBCO XML Transform [33] est une solution de conception complète qui rassemble les fonctionnalités de conception XML et permet de traiter des documents de manière visuelle grâce à la création et au déboguage des feuilles de style XSLT (Extensible Stylesheet Language Transformation). Il offre une interface utilisateur graphique pour l'édition de feuilles de style XSLT qui utilise des mappers simples à base de " glisser-déplacer " pour faciliter le codage des feuilles de style XSLT complexes. Ces feuilles de style peuvent être ensuite intégrées à un environnement d'exécution (" runtime ") pour transformer en toute transparence des documents et des messages XML. L'interface utilisateur de XML Transform fournit également plusieurs fenêtres pour consulter et éditer le code source XSLT, les erreurs de transformation et les résumés des modèles XSLT, ce qui simplifie et facilite le processus de développement.

Le Mapping dans Tibco XML Transform est réalisé d'une manière visuelle par un drag et drop entre schéma ou instance source et cible. Pour de meilleurs résultats, les Mapping drag et drop doivent être spécifiés en utilisant l'approche « top-down ». Le nœud source doit être mappé en premier. Dans les cas où les nœuds père et fils d'un élément à l'intérieur de l'input du document sont mappés vers l'output, le nœud père est mappé en premier.

On peut, par ailleurs, faire un drag et drop entre deux instances de documents ou deux schémas de documents ou une instance et un schéma. Une fois le drag et drop réalisé, une ligne indiquant le Mapping est crée automatiquement dans GraphePane et la couleur bleue indique que l'élément ou attribut et la valeur correspondante est créée (figure 7).

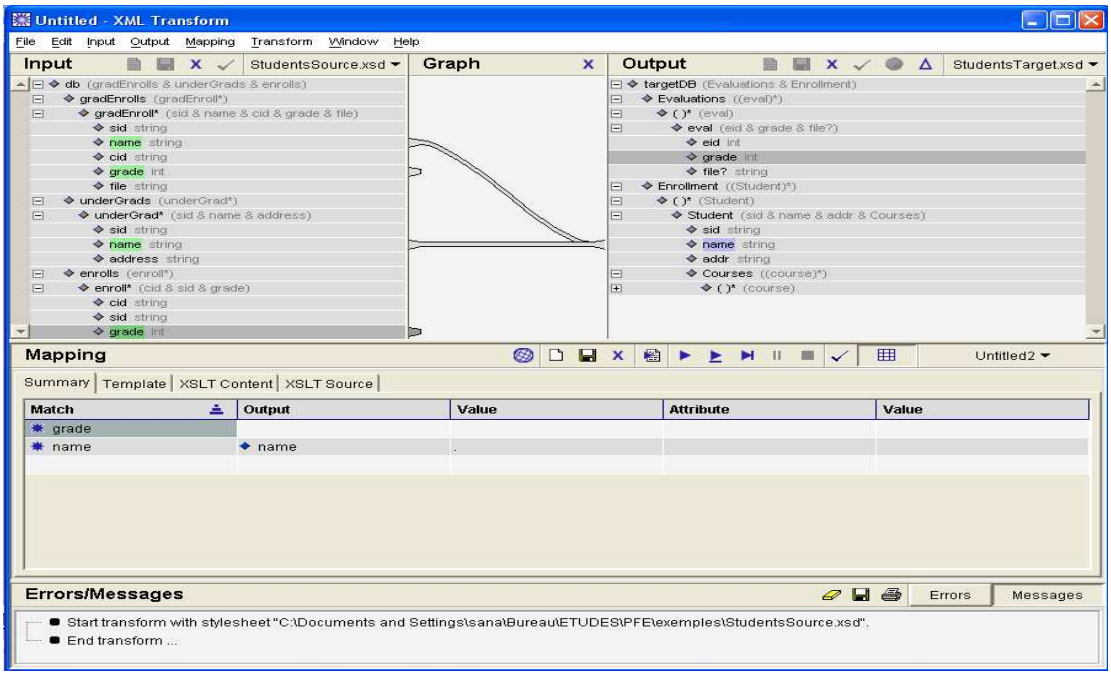


Figure 7 Mapping dans Tibco XML Transform

Limites :

- Un espace de Mapping réduit donc le risque de confusion entre les liens de Mapping.
- Se limite à des transformations XSLT.

2.6 Adeptia XML Mapper

C'est un produit pour manager des règles de transformation des données et un outil graphique qui permet un Mapping visuel des données pour une génération automatique du code de Mapping en XSLT [36].

Il offre à l'utilisateur la possibilité d'utiliser une interface drag et drop pour spécifier les règles de transformations complexes des données. Les règles de Mapping sont définies graphiquement (figure8) et les correspondances XSLT sont générées automatiquement.

Le Mapping est donc défini d'une manière visuelle en joignant les lignes de connexion entre la source et cible. Le Mapping graphique affiche la structure du XML source et donnée cible et montre de façon visuelle les données mappées. Il est maniable et offre un Mapping rapide et clair des structures de données complexes. Il simplifie aussi les règles de management du Mapping car ces règles sont définies et utilisées d'une manière visuelle, spécifiées et faciles à comprendre, stockées pour une autre utilisation, directement converties dans un programme de code exécutable. Il permet donc de manager, réviser, éditer et mettre à jour les règles de Mapping facilement. Il automatise, en plus, la création des transformations complexes du code et minimise le besoin en temps et effort de programmation et améliore la consistance et réduit les erreurs puisque toutes les données de transformation du code sont générées automatiquement. Ce qui permet une réutilisation sur différentes plateformes et environnements.

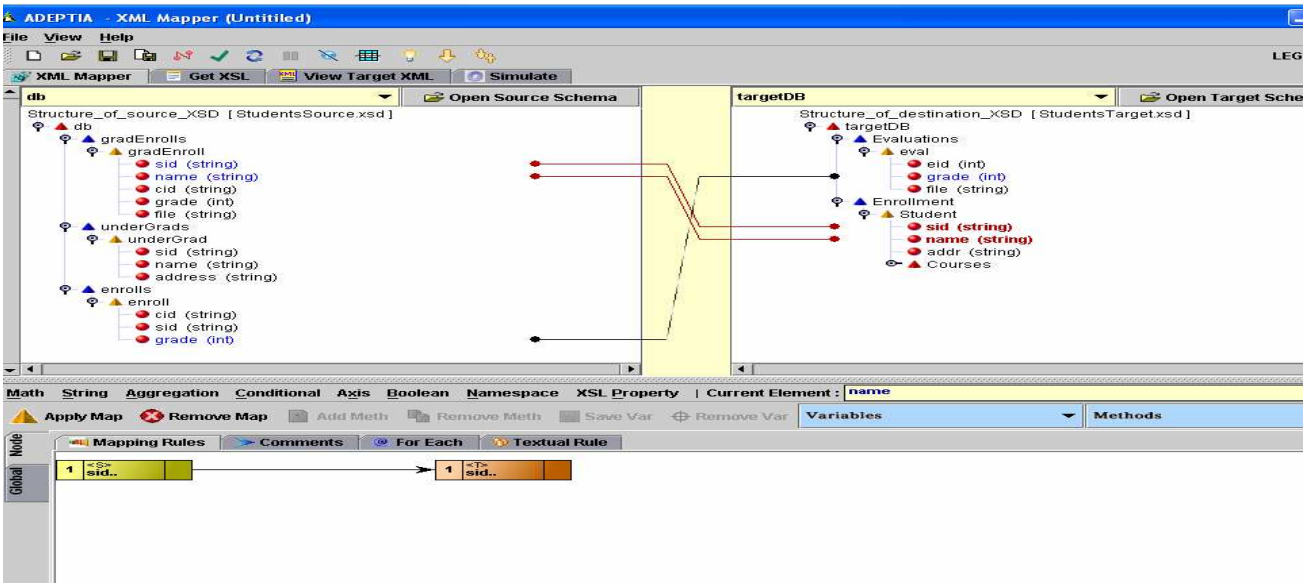


Figure 8 Mapping dans Adeptia XML Mapper

Limites :

- Il se limite à une génération du code XSLT uniquement
- Il n'offre pas la possibilité de "mixer" plusieurs sources et plusieurs cibles pour permettre le Mapping à partir de toutes les combinaisons de format possibles

2.7 Redix AnyToAny XML GUI Mapper

C'est un produit commercial de Redix International [34]. En collaboration avec les outils EDI, il permet de mapper des données entre les messages XML et d'autres messages sans que ça soit un message EDI de base. Une fois les données mappées, cet outil génère les fichiers de Mapping qui seront utilisés par Redix XML conversion pour une transformation en temps réel.

Le Mapping (figure9) est réalisé par des opérations drag et drop dans une Interface utilisateur graphique pour entretenir les conditions de Mapping et les critères de conversion.

Il offre plusieurs fonctionnalités telles que la génération automatique des interfaces déclarées en Visual Basic, Visual C++/C, C++/C ou JAVA, exportation et importation des maps, la personnalisation des messages XML et leur génération automatique.

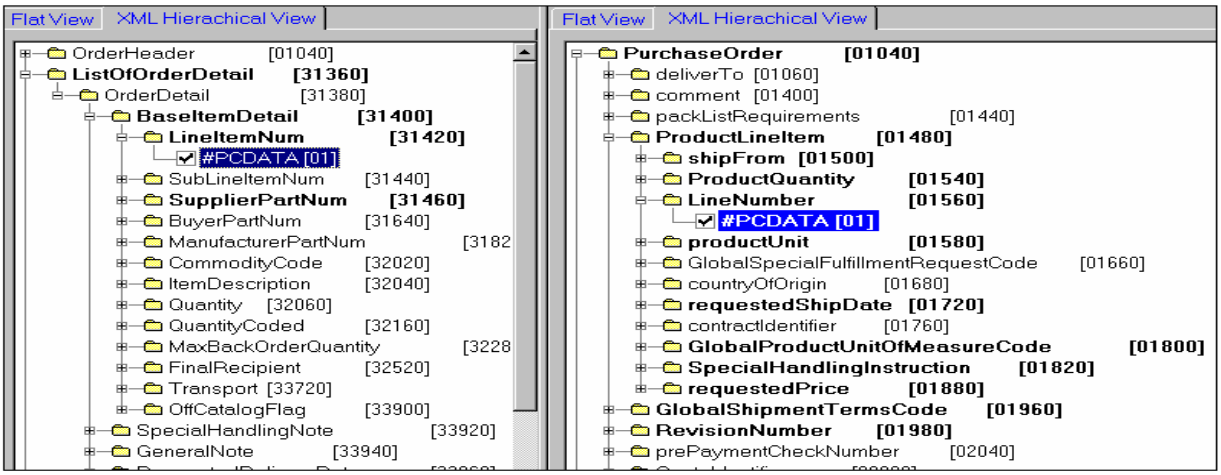


Figure 9 Mapping dans Redix AnyToAny XML GUI Mapper

2.8 Clio

C'est un outil semi-automatique pour la création de schémas de Mappings. Il emploie un Mapping qui dépend de l'utilisation des valeurs de correspondances décrivant quelle valeur de l'attribut cible peut être créée par un ensemble de valeurs d'attributs source (figure 10) [29].

Clio utilise un raisonnement autour des requêtes à créer, manager et classer les Mappings possibles. Cependant, le choix final de Mapping revient à l'utilisateur qui doit comprendre la sémantique de l'application cible [12]. Il génère donc un ensemble de requêtes qu'il transforme et intègre des données de ces sources conformes aux schémas cibles.

C'est un système pour manager et faciliter les tâches complexes des transformations des données hétérogènes et l'intégration. Il supporte la génération et le management des schémas, les correspondances et les Mappings entre les schémas. Il s'intéresse de ce fait à la découverte d'expressions de mappage qui sont les requêtes SQL transformant les données sources en données cibles.

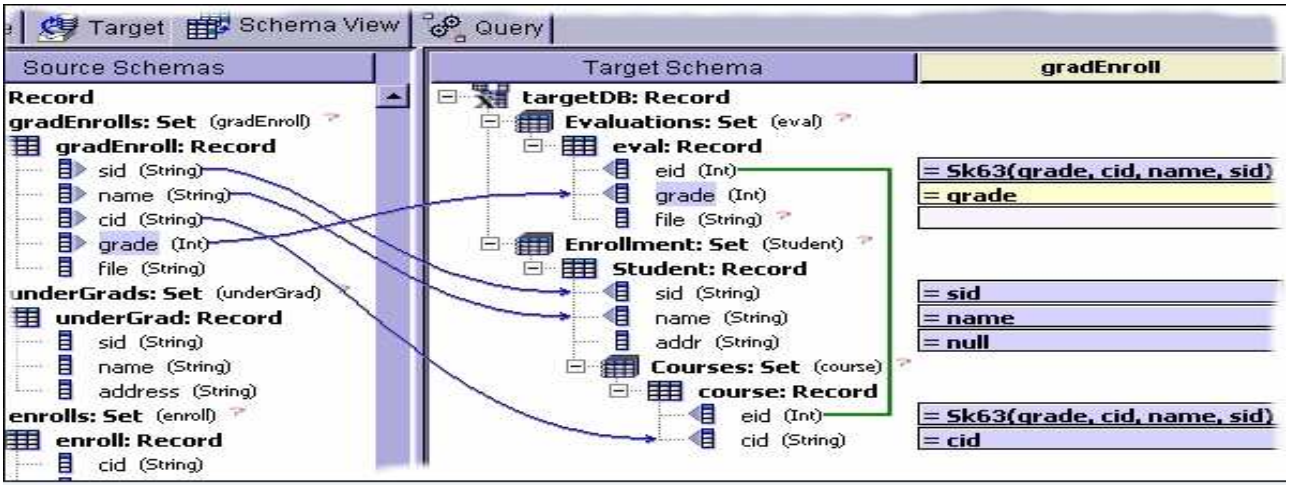


Figure 10 Mapping dans Clio

Limites :

- Clio n'est pas performant dans l'intégration des schémas
- Il ne permet pas un Mapping par drag et drop. En effet, on fait entrer la valeur de correspondance entre l'attribut source vers l'attribut cible pour réaliser le Mapping et une ligne bleue s'affiche entre les deux éléments pour indiquer cette valeur de correspondance.
- Le système clio supporte uniquement les schémas XML et les bases de données relationnelles et pas de sources RDF par exemple [14].

2.9 HyperMapper

HyperMapper [3] est une solution toute récente dont le but étant de fournir à l'utilisateur un ensemble d'outils l'assistant lors des opérations de Mapping de la désignation des fichiers source et cible jusqu' à la transformation du contenu vers le document d'output (figure11).

Il se base sur deux outils :

• ***MapperApplication :*** C'est l'application de Mapping visuel qui renvoie des fichiers de Mapping correspondant respectivement au fichier source et au fichier cible. En outre, elle sera chargée de générer le fichier de transformation XSLT à partir de deux fichiers de Mapping.

• ***MappingEngine :*** Crée les documents XML résultants de la transformation à partir de documents sources XML. MapperApplication est l'application qui débute l'opération de Mapping en créant une représentation visuelle de deux fichiers XML (source et cible) mettant en relief la structure noeuds et attributs de ces derniers.

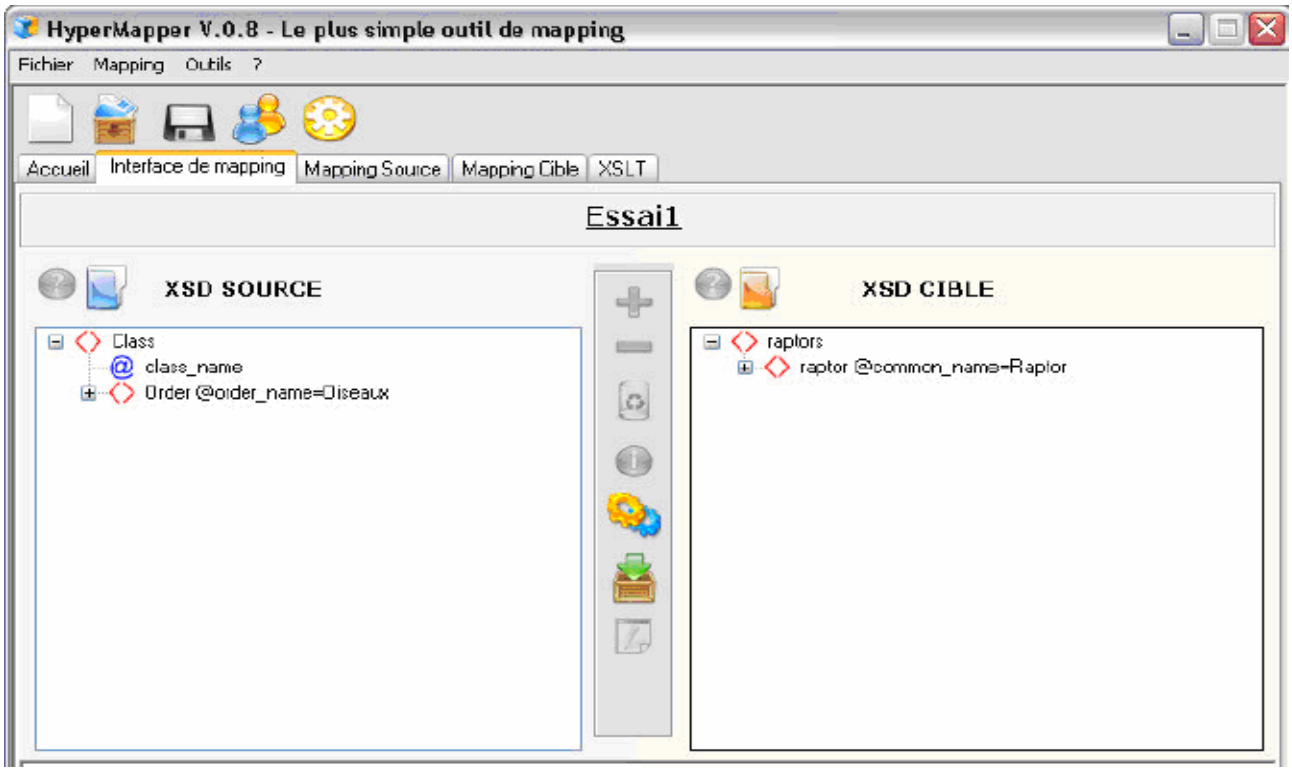


Figure 11 Interface de Mapping dans HyperMapper

Limites :

- Le temps de transformation consommé est long
- Les fonctionnalités sont très limitées.

2.10 BEA WebLogic Workshop

C'est un environnement de développement pour BEA. Sa première application était destinée aux services Web. Il offre un mécanisme de Mapping visuel facile par drag et drop entre les schémas XML et les objets java (figure12). Le Mapping entre deux schémas consiste en une simple connexion de lignes de la source vers la cible [30]. De plus Xquery est représenté en tant que standard pour relier XML au code Java et offre un éditeur visuel complet pour réaliser ces Mappings. On peut donc accéder au code source XQuery en cliquant sur l'icône XQuery tab. Chaque action de visualisation de Mapping est traduite en une expression de Matching XQuery.

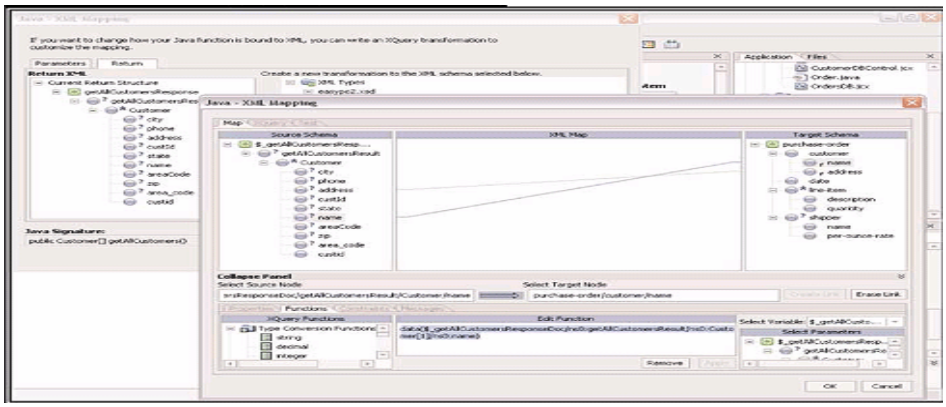


Figure 12 Mapping dans BEA WebLogic Workshop

Limites :

- Il se limite au Mapping entre schéma XML et code java
- Il ne permet pas la génération de code.

2.11 Cape Clear

C'est un produit commercial qui contient l'outil XSLT Mapper [35] qui offre une interface graphique permettant de définir des transformations entre documents XML comme les schémas XML, les DTD et les messages SOAP dans les fichiers WSDL. XSLT Mapper génère, cependant, les transformations XSLT ce qui facilite d'échange de données entre les

organisations. Il permet de faire un Mapping direct entre les nœuds et un Mapping pour les nœuds avec des instances multiples (figure13). Le Mapping des nœuds XML consiste en un drag et drop en sélectionnant le nœud source et cliquant sur le nœud cible. Après le Mapping, XSLT Mapper crée dynamiquement XSLT, qui peut être déployé pour automatiser les transformations des fichiers XML dans l'environnement de développement.

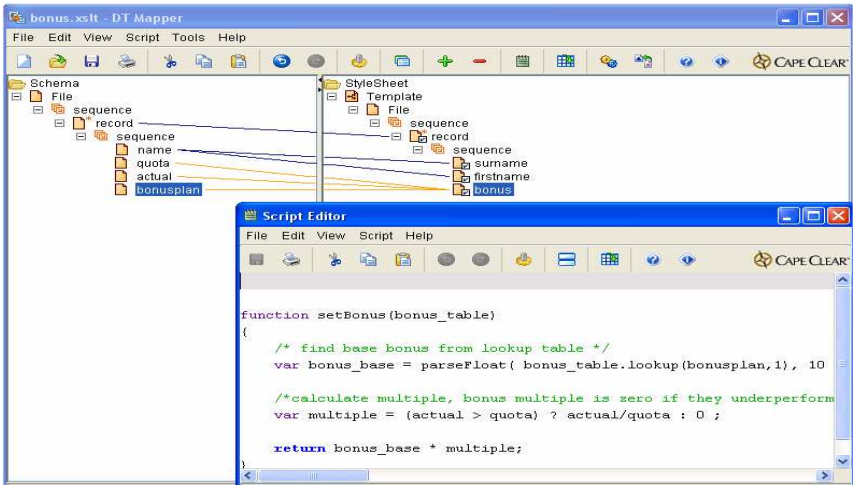


Figure 13 Mapping dans Cape Clear

Limites :

- Il ne permet pas la génération de code
- Il ne permet pas de mapper plusieurs schémas

3. Tableau comparatif

Le tableau suivant est une évaluation globale des différents outils étudiés et existants sur le marché.

	MapForce	**Schema Mapper**	**Stylus Studio**	**Visual XSLT**	**Tibco**	**Adeptia XML Mapper**	**BEA Weblogic**	**Clio**	**Cape Clear**
Type de données en entrée	Fichiers XML, BD, Données EDI	Schémas XML	Schémas XML, DTD, instances de docuements XML	Schémas XML ou DTD	FichiersXML, schémas/DTD	DTD, schémasXML	Schémas XML et objets java	Schémas XML ou instances DB2	Schémas XML, DTD, messages SOAP
Codes générés	XSLT 1.0 XSLT 2.0 XQuery Java, C++ ou C#	XSLT	XSLT et XQuery	XSLT	XSLT	XSLT	Java	XQuery	XSLT
Transformations XSLT	Oui	Oui	Oui	Oui	Oui	Oui	Non	Non	Oui
Interface graphique	Conviviale par drag et drop	Mapping par simple clic	drag et drop	drag et drop	drag et drop	drag et drop	drag et drop	Non conviviale	Mapping par drag et drop
Mapping entre nœuds et attributs	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui	oui
Mapping de plusieurs sources vers une cible	Oui mais avec duplication du nœud ou avec utilisation de fonctions prédéfinies	Non juste entre deux noeuds	Oui en associant une fonction	On peut mapper plusieurs noeuds	Non	Non	Non	Oui	Oui
Edition de schémas	Non	Oui	Oui	Non	Oui	Oui	Oui	Non	Oui
Mapping de plusieurs schémas	Oui	juste des schémas XML	non	Non	Non	Non	Non	Non	Non

Tableau 2 Tableau de comparaison des outils de Mapping

- Type de données en entrée: Ce critère permet de juger la généricité de l'outil. Plus l'outil permet le Mapping entre différents formats de données plus il est performant.
- Codes générés : Tous les codes qui peuvent être générés automatiquement par les différents outils de Mapping.
- Transformation XSLT : Vérifier si tous les outils assurent la fonction de transformation de XSLT.
- Interface graphique : Peut être caractérisée par un Mapping drag et drop mais aussi par d'autres moyens ainsi la présentation graphique de l'interface inclut la surface de Mapping, la disposition des schémas, la clarté de la visualisation.
- Mapping de plusieurs sources vers une cible : Ce critère permet de vérifier si tous les outils peuvent assurer la fonction de Mapping de plusieurs attributs vers un autre.
- Edition de schémas : Les outils peuvent disposer ou pas d'éditeurs de schémas.
- Mapping de plusieurs schémas : Ce qui distingue un outil par un autre c'est la possibilité de mapper plusieurs schémas sur une interface unique ainsi le Mapping ne sera pas réduit à deux schémas.

4. Conclusion

Les outils de Mapping représentent plusieurs fonctionnalités. Certains sont plus performants que d'autres. Notre motivation vient du fait que tous les outils existants sur le marché ne réalisent pas en même temps le matching, le mapping et la génération de code. L'idée est de réaliser un outil qui permet d'intégrer toutes ces spécificités. Un outil simple qui soit simple d'utilisation, avec une interface flexible, conviviale, d'implémenter le prototype GUI (Graphical User Interface), utiliser le drag et drop des objets représentant des schémas, des fonctions de Mapping, des conditions de Mapping, des schémas multiples…Générer différents codes tels que Xquery, java.. .

Le chapitre suivant aura pour objectif de décrire toutes ces spécificités au sein d'une seule architecture.

Chapitre 5 Architecture ASMADE

1. Introduction

La première motivation de ce travail est de fournir une plateforme qui enchaîne les processus de Matching et de Mapping afin de minimiser les interventions humaines. Pour parvenir à réaliser le Matching et les transformations entre les schémas XML, nous proposons l'architecture ASMADE (Automated Schema Matching For Documents Exchange) permettant d'offrir un cadre d'application pour automatiser les transformations des instances des schémas.

2. Présentation de l'architecture ASMADE

L'architecture du système ASMADE (Automated Schema Matching For Documents Exchange) se décline en quatre couches (Matching, Filtering, Mapping, Transformation). Les entrées sont transformées en des arbres par l'algorithme EXSMAL. Le processus de Matching offre un ensemble de correspondances (1-1, 1-n, n-m). Le processus de Filtrage est ainsi représenté afin d'éliminer les correspondances non pertinentes. Le résultat de cet ensemble de correspondances est ainsi exprimé dans le modèle XME (XML Mapping Expression) pour saisir les Mappings nécessaires. Finalement, ces Mappings seront générés en XQuery et exécutés par un moteur d'exécution XQuery (figure14).

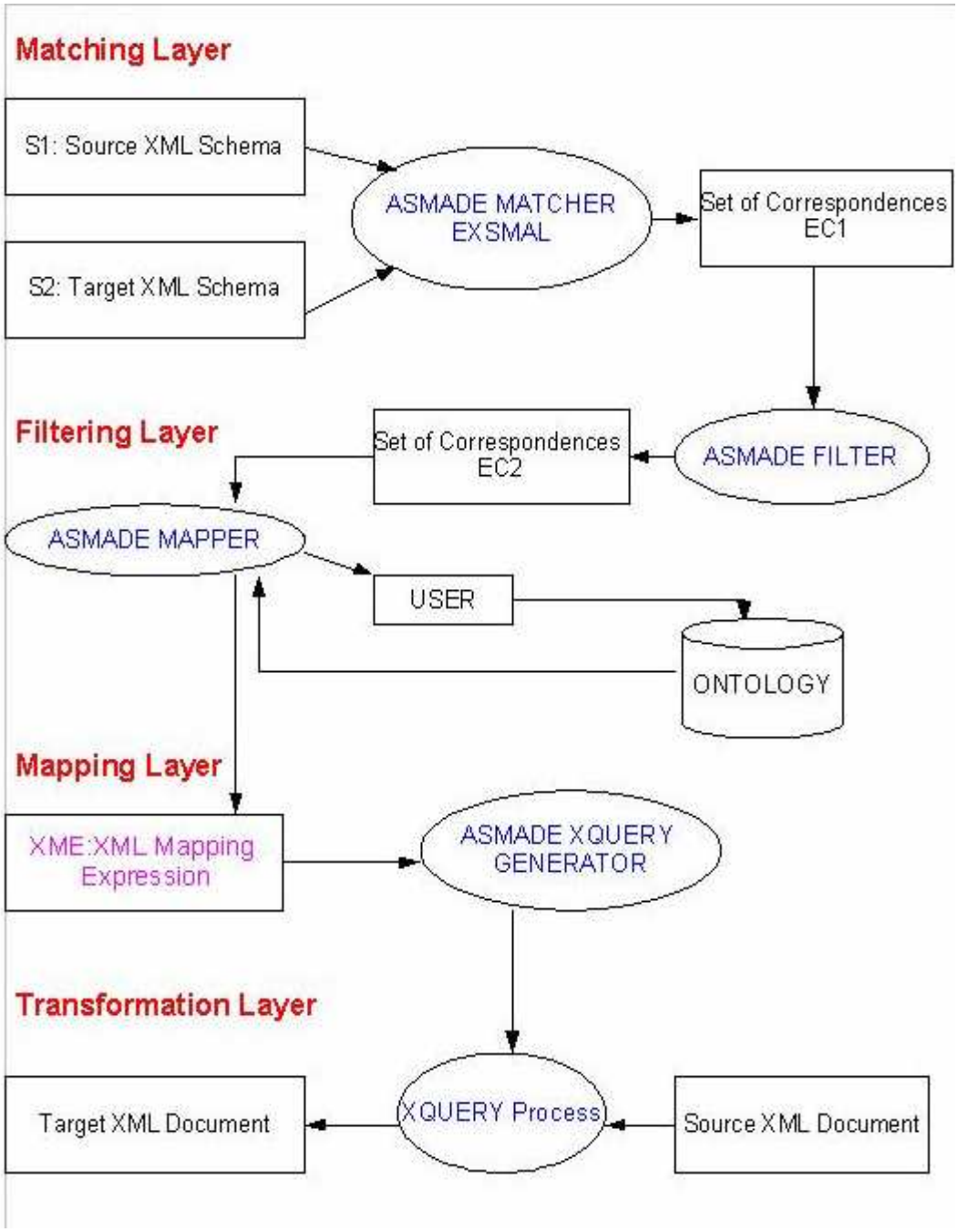


Figure 14 Architecture de ASMADE

2.1 La Première Couche: Matching Layer

Les entrées de ce cadre d'application est un couple de schémas XML (S1 qui représente le schéma source et S2 qui représente le schéma cible) qui sont transformées en arbres traités par l'algorithme EXSMAL qui calcule l'ensemble des correspondances EC1 [4].

2.2 La Deuxième Couche: Filtering Layer

La seconde couche permet de filtrer les Matchings produits par EXSMAL en éliminant au maximum les correspondances non pertinentes entre les éléments des schémas XML. Pour réaliser ces calculs[25], deux nouveaux calculs de similarité de base et structurelle ont été proposés qui sont le calcul de la similarité de chemin au niveau des nœuds feuilles pour extraire les relations de Matching 1-1 au niveau de ces nœuds. Ensuite, le calcul de la similarité interne au niveau des nœuds internes pour extraire les relations de maching de cardinalité 1-1 au niveau de ces nœuds.

2.3 La Troisième Couche : Mapping Layer

Dans cette troisième couche, il s'agit d'utiliser le résultat précédent désigné par EC2 sur la figure pour générer des règles de transformation utilisables. Théoriquement, ces règles sont exprimées dans le modèle XME (XML Mapping Expression).

Dans ce modèle, nous respecterons la spécificité du modèle d'expression de mappage défini dans [1] en précisant davantage la génération des éléments/attributs cibles. Nous étendons donc le modèle présenté dans le chapitre3 paragraphe 2.2 pour définir un attribut comme suit:

$$Ai = (\overline{\sum}_i \; [f_i(a_r, a_p, \ldots, a_e), l_i, c_i] + \tau_i = \alpha_i(A) + \tau_i$$

Équation 5 Modèle d'expression de mappage étendu

- α_i: l'expression qui permet de générer une partie de l'attribut/champ cible.

$$\alpha_i = \overline{\sum}_i [f_i(a_r, a_p, \ldots, a_e), l_i, c_i]$$

- $\overline{\sum}_i$ représente la fonction de concaténation
- ***f_i* :** est la fonction de mappage qui peut être arithmétique ou n'importe quelle fonction composée de manipulation de chaînes de caractère (ex : substring). Alors f_i peut être appliquée sur un ensemble d'attributs/champs qui appartiennent à une ou plusieurs sources. On peut définir *Attribut (f_i)* = { a_r^{sr} , $a_p s^p$,...., a_e^{se} } où $a_r \in S_r$, a_r est un attribut/champ de S_r , comme l'ensemble des attributs sources qui interviennent dans l'expression du mappage.

- ***L_i* :** ensemble de filtres sur les données sources. On peut identifier un filtre pour chaque attribut/champ a_r alors $l_i = \{l_i (a_r) / a_r \in S_r\}$. Ce filtre peut tester la valeur de l'attribut/champ a_r ou bien la valeur d'un ou plusieurs attributs/champs de la même source.
- ***C_i* :** défini comme étant la condition à satisfaire avant d'effectuer le mappage. Elle nous permet d'ignorer certaines données si elles ne satisfont pas la condition.
- **τ_i:** GenericFunction: fonction générique, complètement indépendante de la source. Elle permet de couvrir les cas que le modèle d'expression de mappage défini dans [1] ne peut pas générer. Elle peut être une valeur constante, ou une fonction incrémentale, une fonction automatique, une fonction de skolem.

2.3.1 Représentation XSD de XME

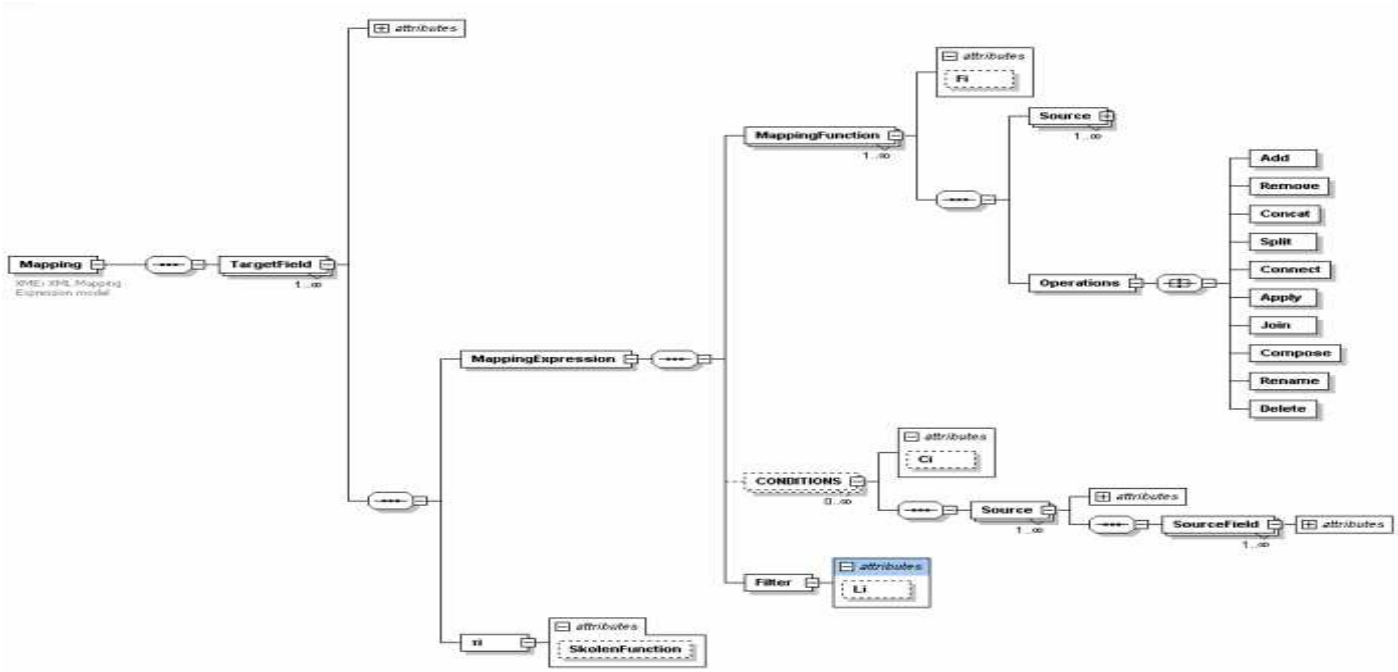


Figure 15 Représentation de XME

2.3.2 Instance du modèle XME

```
<Mapping>
< TargetField   Name= « Ai »>
  <MappingExpression>
     <Function Name="fi">
          < Source Name="Si">
                <SourceField Name="ai"/>
                    ...
                <SourceField Name="an"/>
          </Source>
        < Source Name="Sj">
                <SourceField Name="ai"/>
                    ...
                <SourceField Name="an"/>
          </Source>
    </Function >
    <Condition Name="Ci">
          < Source Name="Si">
                <SourceField Name="ai"/>
                    ...
                <SourceField Name="an"/>
          </Source>
        < Source Name="Sj">
                <SourceField Name="ai"/>
                    ...
                <SourceField Name="an"/>
          </Source>
    </Condition>
    <Filter Name="Li">
    </Filter>
</MappingExpression>
</TargetField>
<τi>
<Function Name="GenericFunction"/>
</ τi>
</Mapping>
```

2.3.3 Opérateurs de transformations

L'identification des correspondances entre les différents schémas hétérogènes doivent être validés par l'expert du domaine. Ceci nous permettra d'appliquer les primitives de transformation entre les entités des schémas (source et destination) et donc d'employer des opérateurs de transformation réalisant la tâche de Mapping proprement dite. Une liste d'opérateurs de transformations doit être définie pour couvrir les différents cas ou situations de Mapping facilitant ainsi la génération du Mapping entre les entités.

A) L'opérateur Add:

Cet opérateur décrit une entité qui apparaît dans le schéma cible mais qui n'est pas dans le schéma source.

***B) L'opérateur* Remove**

Cet opérateur décrit une entité ou type dans le schéma source qui n'apparaît pas dans le schéma cible.

***C) L'opérateur* Concat**

Cet opérateur décrit La valeur du champ cible est établie par concaténation des valeurs du champ source.

***D) L'opérateur* Split**

Cet opérateur est l'opération inverse de Concat.

***E) L'opérateur* Connect**

Cet opérateur correspond à un Matching 1-1 qui relie deux entités équivalentes sans aucune transformation. Cet opérateur décrit une entité ou type à partir d'un schéma source qui est totalement identique à l'entité ou type dans le schéma cible. Etre identique dans cette forme requiert que tous les attributs noms et types soient identiques.

***F) L'opérateur* Apply**

Cet opérateur prend un élément/attribut et une fonction arbitraire f qui peut être arithmétique comme inputs et applique f sur cet élément. La valeur du champ cible est le résultat de cette fonction sur les valeurs du champ source.

***G) L'opérateur* Join**

Cet opérateur permet de joindre ou combiner des entités appartenant à des schémas sources

différents et produit un nouvel attribut comme étant en une concaténation des ces deux entités afin de la mapper avec l'attribut cible.

***H) L'opérateur* Compose**

Le champ source peut avoir deux valeurs. Le champ cible dépend des valeurs de ce champ. Par exemple, la valeur du champ source X=1 représente Enr1 dans la cible et X=2 représente Enr2 dans la cible.

***I) L'opérateur* Rename**

Cet opérateur décrit une entité ou type à partir d'un schéma source qui est totalement identique à l'entité ou type dans le schéma cible comme décrit au-dessus à part le nom de l'entité ou type qui change.

***J) L'opérateur* Delete**

Cet opérateur permet de supprimer les attributs qui ne participent pas au Mapping.

K) Exemple d'instance de la Représentation XME

Ceci est un exemple de représentation XME de la fonction de concaténation.

```
<Mapping>
< TargetField   Name= « Name »>
  <MappingExpression>
      <Function Name=”Concat”>
          < Source Name=”S1”>
                <SourceField Name=”FirstName”/>
          </Source>
        < Source Name=”S1”>
                <SourceField Name=”LastName”/>
          </Source>
< /TargetField >
</Mapping>
```

2.4 Quatrième Couche : Transformation Layer

Dans cette dernière couche, Les Mappings résultants seront traduits dans le langage de requêtes XQuery. Notre choix s’est rapporté au langage XQuery et non pas à un autre langage tel que XSLT par rapport à sa facilité d'utilisation, la capacité d'optimisations, le typage de données fort, les Mappings de XML à XML, de XML à non XML et du non XML à XML et la génération des vues

dans l'intégration des données. De plus, transformer ou requêter des informations techniques typées devrait ne se faire qu'en XQuery. Pour traduire ce Mapping en XQuery, chacun des Mappings sera transformé selon l'expression FLWR (For- Let-Where-Return) [37].(Annexe B)

3. Conclusion

Lors de ce chapitre, nous avons proposé une architecture ASMADE, une approche de Matching de schémas, pour l'échange des données. Dans le chapitre qui suit, nous allons réaliser la conception de cette architecture.

Chapitre 6 Conception

1. Introduction

Lors de ce chapitre, nous allons identifier les diagrammes de classes et de cas d'utilisation réalisés pour mettre en œuvre l'architecture ASMADE proposée. La motivation fondamentale de la modélisation est de fournir une démarche antérieure afin de réduire la complexité du système étudié lors de la conception et d'organiser la réalisation du projet en définissant les modules et les étapes de la réalisation. Plusieurs démarches de modélisation sont utilisées. Nous adoptons dans notre travail une approche objet basée sur un outil de modélisation UML.

En fait, UML (*Unified Modleing Language*) est un standard ouvert contrôlé par l'OMG, un consortium d'entreprises qui a été fondé pour construire des standards qui facilitent l'interopérabilité et plus spécifiquement, l'interopérabilité des systèmes orientés objet.

UML est issu de l'unification de nombreux langages de modélisation graphique orientée objet. Il unifie à la fois les notations et les concepts orientés objets.

2. Conception de l'architecture

2.1 Identification des diagrammes

Pour modéliser l'architecture ASMADE, nous allons identifier trois diagrammes:

- Les diagrammes de cas d'utilisations représentent un intérêt pour l'analyse des besoins métier ce qui nous permettra de démarrer l'analyse orientée objet et identifier les classes candidates.
- Le diagramme d'activités décrit la succession des activités et leurs interactions dans un processus du système.
- Un diagramme de classes est une collection d'éléments de modélisations statiques (classes, paquetages...), qui montre la structure d'un modèle. Les classes sont liées entre elles par des associations. Une association permet d'exprimer une connexion sémantique bidirectionnelle entre deux classes.

2.2 Conception des couches

Cette phase consiste à enrichir la description du procédé, de détails d'implémentation afin d'aboutir à une description très proche d'un programme. Nous allons modéliser toute l'architecture en diagramme de cas d'utilisation, d'activités et de classes.

2.2.1. Diagramme de cas d'utilisation

Le diagramme suivant représente le cas d'utilisation de notre architecture (figure16). Il décrit le comportement du système du point de vue utilisateur. En effet, l'utilisateur va choisir les schémas XML source et cible, le système réalise le Matching entre les schémas pour avoir les valeurs de correspondances entre les entités des schémas. Ces valeurs de correspondances seront filtrées, on aura des valeurs de correspondances plus précises. Ensuite, l'utilisateur réalise le Mapping et le résultat sera généré en XQuery.

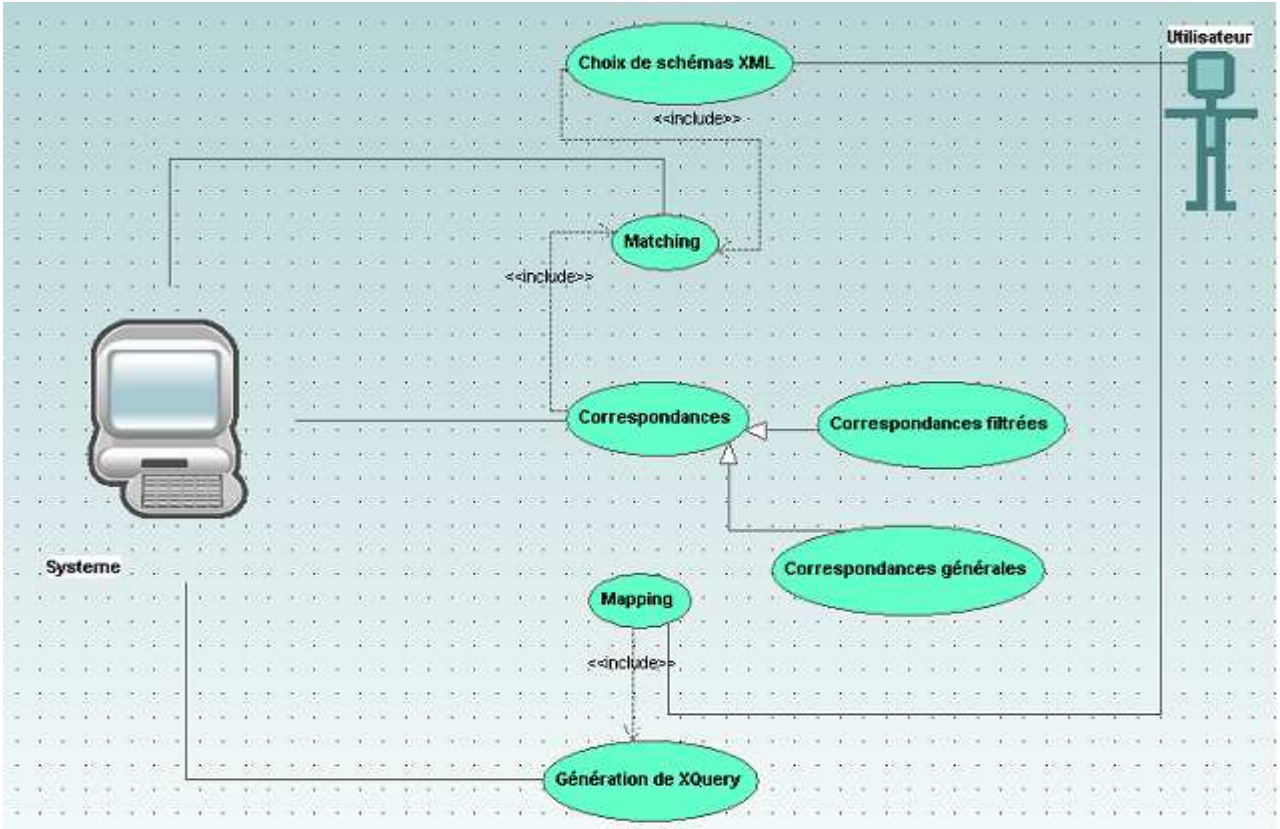


Figure 16 Diagramme de Cas d'utilisation

2.2.2 Diagramme d'activités

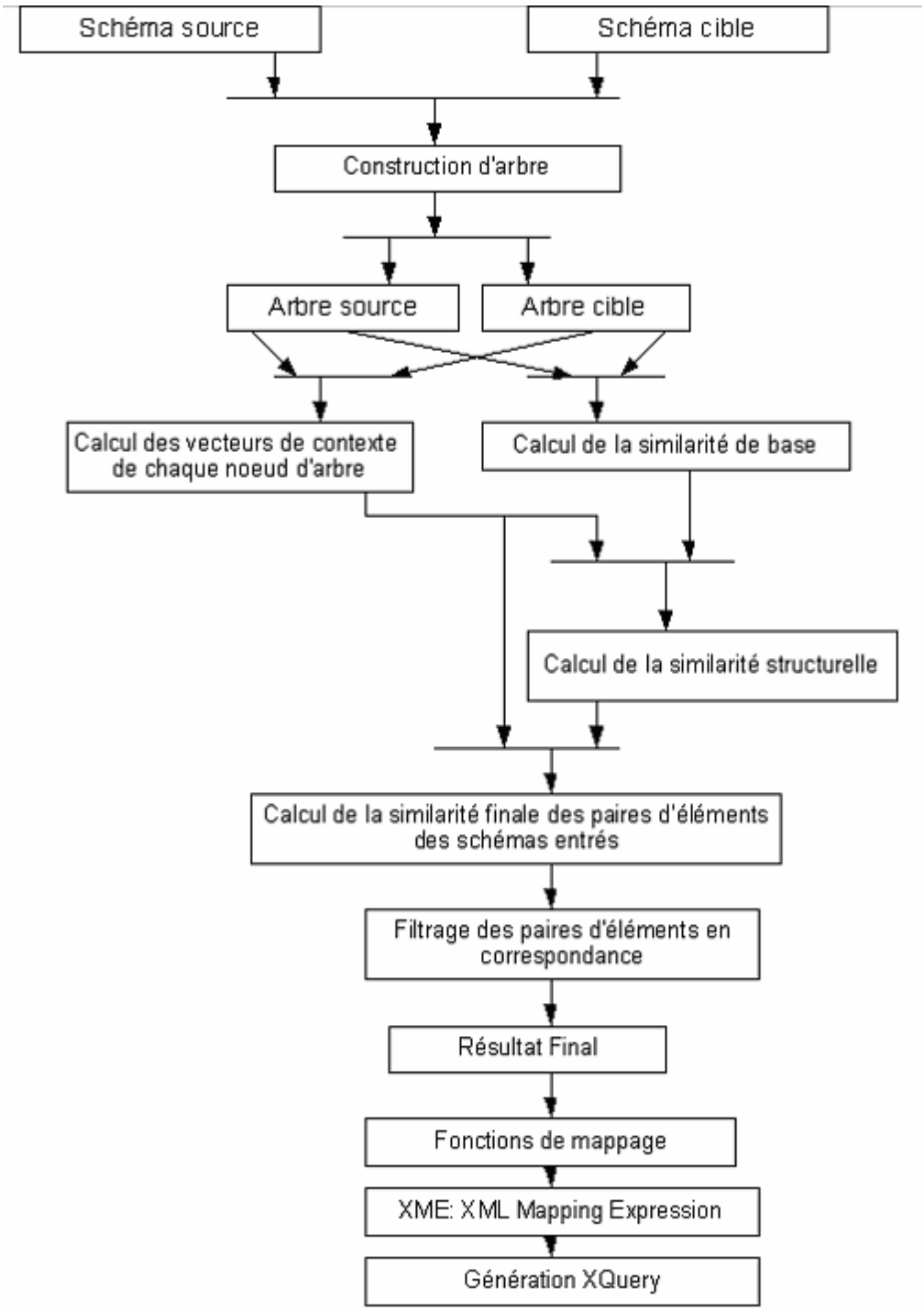


Figure 17 Diagramme d'activités représentant l'intéraction entre les différents modules

Huit étapes sont définies (figure17) :

- ***Première Etape : Construction de l'arbre***

Cette étape consiste à prendre les deux schémas XML (.xsd) en entrée et les convertir en arbre. Pour faciliter notre implémentation, les schémas n'ont pas d'éléments partagés et toutes les références de groupe d'attributs sont remplacées par leurs définitions. Nous traitons les éléments, les groupes d'attributs et les attributs de la même manière car ils ont tous les descriptions textuelles et les types de données associés. Ils sont convertis en nœuds lors de la phase de construction de l'arbre.

- ***Deuxième Etape : Calcul de similarité de base***

Cette étape prend en entrée deux arbres dont les nœuds sont spécifiés précédemment et calcule la similarité entre les nœuds individuels de chaque arbre en se basant sur leurs descriptions textuelles et leurs types de données.

- ***Troisième Etape : Calcul des vecteurs de contexte de chaque nœud d'arbre***

Cette étape se déroule en parallèle avec l'étape 2. On parcourt les deux arbres et pour chaque nœud, extrait les valeurs de son contexte, c'est-à-dire, les quatre vecteurs contenants respectivement les nœuds ancêtres, les nœuds frères, les nœuds fils immédiats et les nœuds feuilles du sous arbre en élément en question. Chaque nœud de deux arbres est associé à un contexte de nœud, qui est un objet constituant de quatre vecteurs.

- ***Quatrième Etape : Calcul de la similarité structurelle***

Dans cette étape, le calcul de similarité structurelle entre deux nœuds se fait à l'aide des résultats de similarité de base calculés dans la troisième étape et en prenant en compte des similarités de chaque élément de leurs contextes (ancêtre, frère, fils immédiat et feuille). Dans cette étape, afin de bien traiter chaque contexte de manière flexible et correcte, on devra réajuster les valeurs de coefficients utilisées pour calculer la similarité structurelle avec les 4valeurs provenant de chaque attribut de contexte dans certains cas spéciaux. Les valeurs de similarité structurelle sont stockées pour l'usage dans la prochaine étape.

- ***Cinquième Etape : Calcul de la similarité finale des paires d'éléments des schémas entrés***

Les sorties de la deuxième et quatrième étape font les entrées de cette étape. Pour chaque paire de nœuds, la valeur de la similarité finale est calculée en se servant de leurs valeurs de similarité de

base avec celle de structure. Ces résultats finaux sont stockés pour être utilisé dans la dernière étape.

- ***Sixième Etape : Filtrage des paires d'éléments en correspondance***

Cette dernière étape consiste à choisir parmi les correspondances trouvées dans l'étape précédente celles qui sont les plus plausibles. Pour ce faire, à l'aide d'une valeur de seuil située entre 0 et 1, on élimine les paires d'éléments en correspondance dont la similarité est inférieure à ce seuil. L'ensemble des correspondances après le filtrage constitue le résultat final définitif d'un processus du Matching entre deux schémas.

- ***Septième Etape :Détermintion des Mappings entre les schémas***

A partir des correspondances de Matchings résultants, on définit les fonctions de mappage définis dans le chapitre précédent entre les entités des schémas. Ensuite, le résultat sera présenté en modèle XME .

- ***Huitième Etape: Génération en XQuery***

Le résultat de l'étape précédente sera sauvegardé pour ensuite être généré en XQuery afin de permettre la transformation de documents.

2.2.3 Diagramme de classes

Le diagramme de classes représenté dans la figure18 suivante décrit les associations entre les classes et ceci afin de déterminer les dépendances entre les différentes classes.

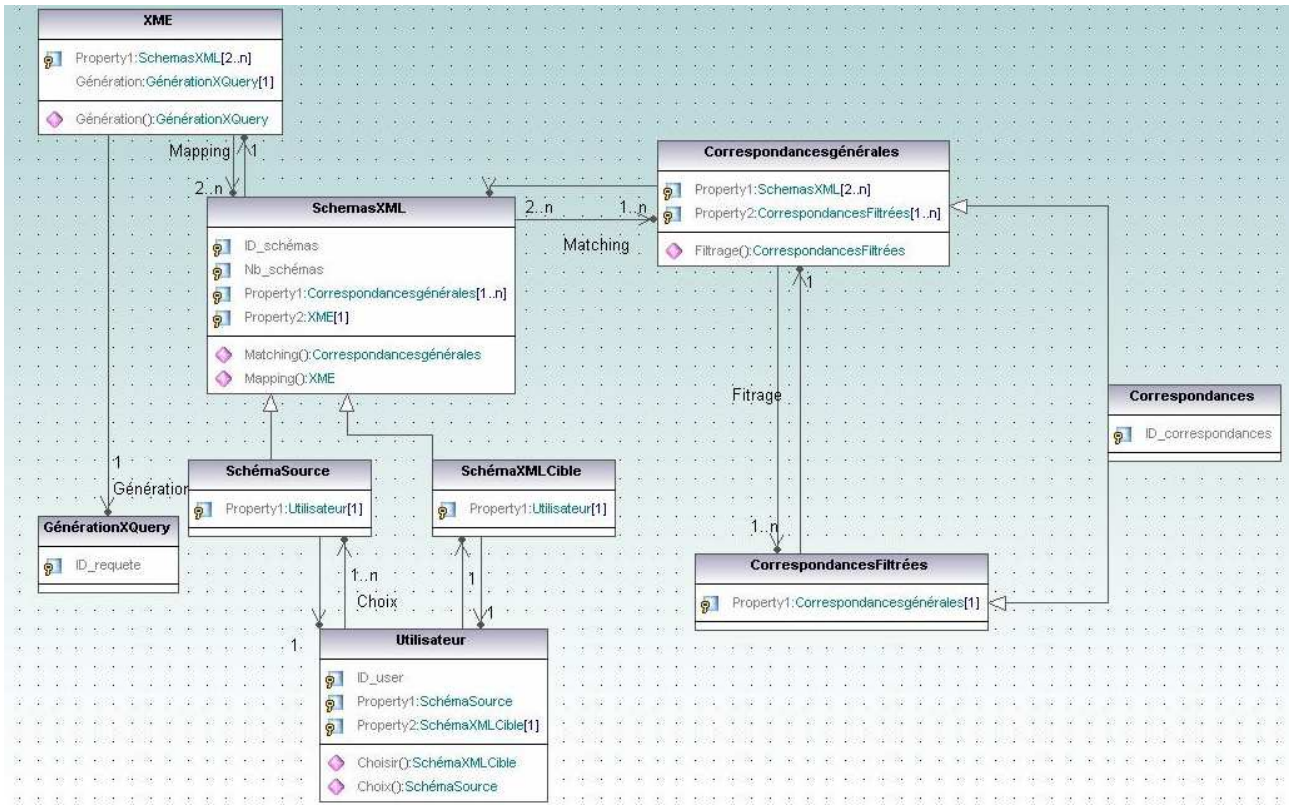


Figure 18 Diagramme de classes

3. Conclusion

Dans ce chapitre, nous avons identifié les diagrammes de cas d'utilisation, d'activités et de classes pour faciliter la réalisation de notre prototype.

Dans le chapitre suivant nous montrerons les étapes, plus en détails, que nous avons suivies pour implémenter et réaliser notre solution.

Chapitre 7 Réalisation

1. Introduction

L'implémentation est la phase la plus importante après celle de la conception. Le choix des outils de développement influence énormément sur le coût en temps de programmation, ainsi que sur la flexibilité du produit à réaliser.

Cette phase consiste à transformer le modèle conceptuel établi précédemment en des composants logiciels formant notre système.

Dans ce chapitre, nous allons commencer par la description de l'environnement de travail puis à dégager et élaborer les composants de notre système.

2. Environnement de travail

L'environnement de travail est constitué par deux parties nommées environnement matériel et environnement logiciel.

2.1 Environnement matériel

Le développement de l'environnement matériel est caractérisé par :

1. Système d'exploitation : Windows XP Professionnel.
2. CPU : Pentium M, 1.6 GHz
3. Mémoire : 512 Mo

2.2 Environnement logiciel

L'environnement logiciel consiste les composants suivants :

- Java 2 Platform, Enterprise Edition J2EE JDK 1.5
- Outil de développement Eclipse

- Outil Altova UModel
- L'utilisation du Serveur cvs (Concurrent Versions System) qui est un serveur de partage des ressources. Comme l'équipe est partagée entre les Etats-Unis et la France, on a eu recours au serveur pour avoir accès au code, apporter les modifications, des améliorations, y ajouter des commentaires, etc…

3. Implémentation

3.1 Choix de langage de programmation : Java

Pour implémenter notre système, le langage de programmation java est le mieux adapté. En effet, Java s'annonce comme une des évolutions majeures de la programmation. Pour la première fois, un langage efficace, performant, standard et facile à apprendre (et, de plus, gratuit) est disponible.

Il est un langage multi-plate-forme qui permettrait, selon le principa1 proposé par Sun Microsystems, son concepteur, d'écrire des applications capables de fonctionner dans tous les environnements.

L'objectif était de taille, puisqu'il a réussit à gagner une grande popularité auprès des programmeurs grâce à ses avantages. Ce langage offre une portabilité maximale grâce à une indépendance totale par rapport au système.

3.2 Développement de l'application

Dans cette partie, nous allons présenter les différentes phases de la réalisation de notre projet en mentionnant des imprimes écrans de notre application.

3.2.1 Prototype

Le prototype réalisé s'identifie par une interface (figure 19) où sont définies toutes les étapes à réaliser. Un ensemble de boutons est défini à gauche dans l'interface pour identifier et choisir les schémas source et cible, un autre bouton pour les fonctions de Mapping, pour le Matching et le grand panel à droite, panel central, où sont réalisées toutes les manipulations.

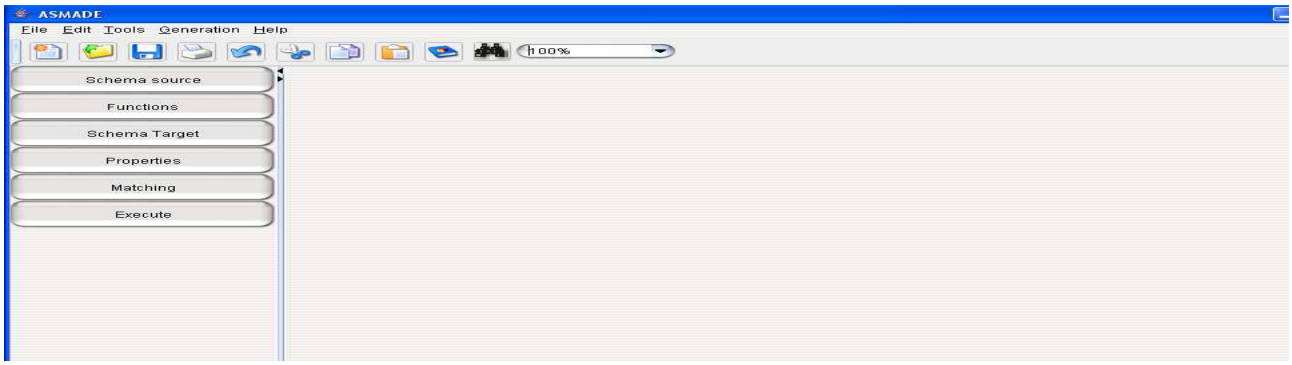


Figure 19 Interface de ASMADE

3.2.2 Choix des schémas

La spécificité de cette interface est la possibilité de choisir plusieurs schémas sources (en cliquant sur Schema source) qu'on peut matcher et mapper avec le schéma cible (en cliquant sur Schema Target) (figure 20). Cette interface (figure 21) représente une convivialité du fait que les schémas se déplacent dans l'interface et ne sont pas définis en tant que schémas fixes. Les schémas sont représentés sous forme d'arbres.

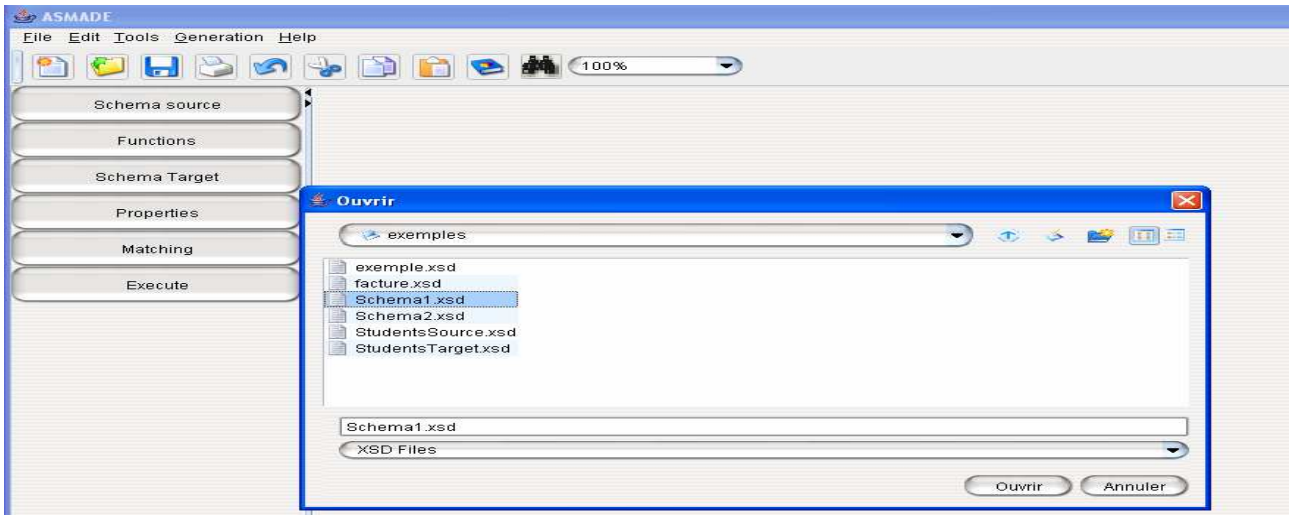


Figure 20 Choix des schémas à comparer

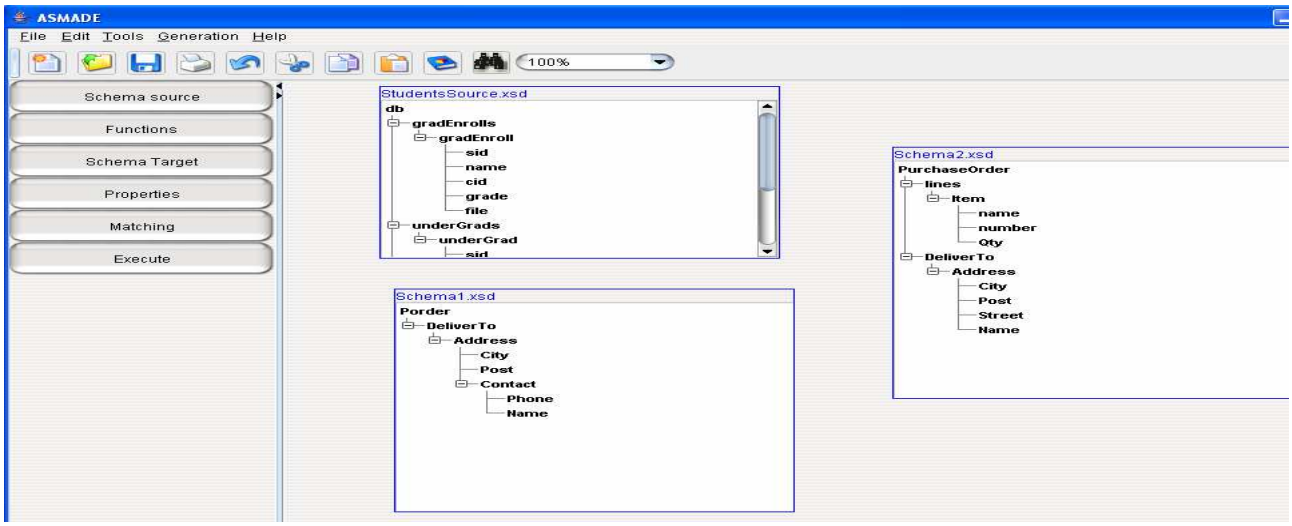


Figure 21 Sélection des schémas

3.2.3 Matching des schémas

Après avoir choisi les schémas à comparer, on clique sur le bouton Matching qui affiche un ***box*** qui contient les différentes valeurs de coefficients (Description Coefficient, Ancestor Node Coefficient, Sibling Node Coefficient, Immediate Child Node Coefficient, Standard Deviation Threshold, Basic Similarity, Acceptation Threshold) (figure22) désirés pour calculer le Matching entre les deux schémas. Les correspondances sémantiques entre les deux schémas à comparer sont liées par des lignes (figure 23).

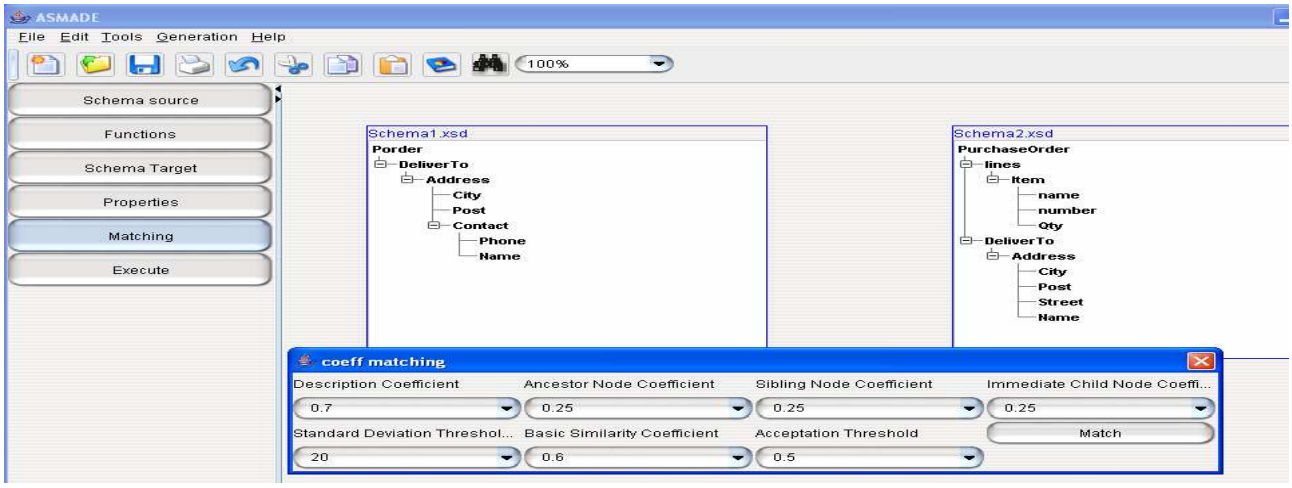


Figure 22 Choix des Coefficients pour le Matching

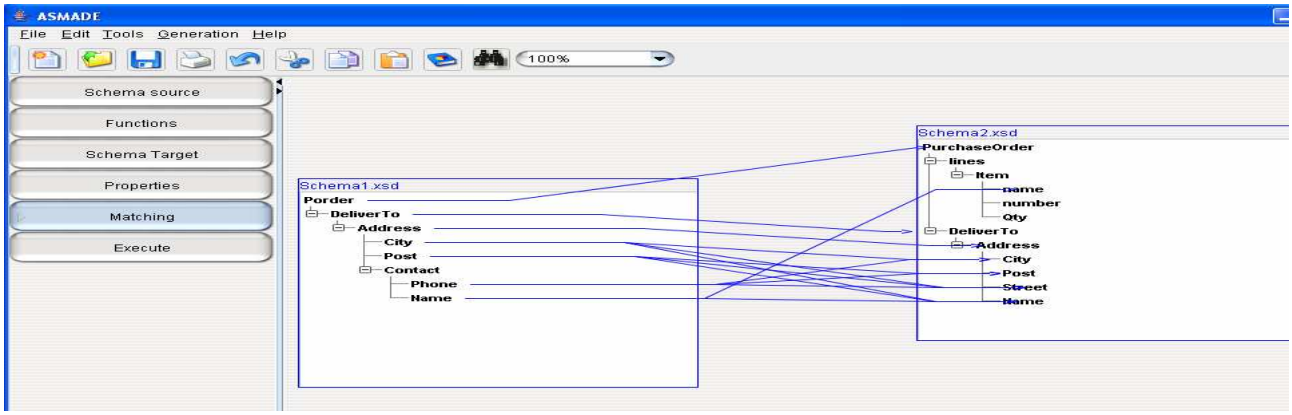


Figure 23 Matching entre les schémas

3.2.4 Filtrage

Grâce à la valeur de “Acceptation Threshold”, on peut agir sur le raffinement des correspondances, soit le nombre de lignes de connexions qui étaient affichées lors de l’étape de Matching. Ceci est lié au fait que le nombre de paires d’éléments en correspondance qui varie selon la valeur de “Acceptation Threshold”. Par exemple, dans la figure 24 suivante la valeur est égale à 0,65 ce qui affine le résultat de Matching.

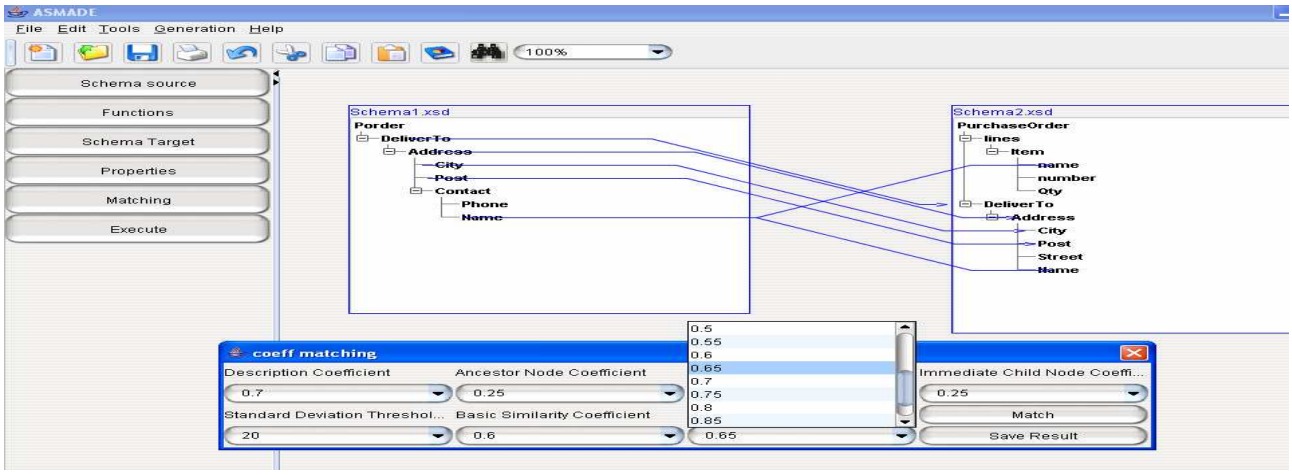


Figure 24 Choix du coefficient de filtrage

3.2.5 Sauvegarde du résultat

Pour sauvegarder le résultat du Matching, il faut appuyer sur le bouton “Save Result”. Le résultat sera stocké sous forme d’un fichier XML qui pourra être restauré pour des prochains usages (figure 25).

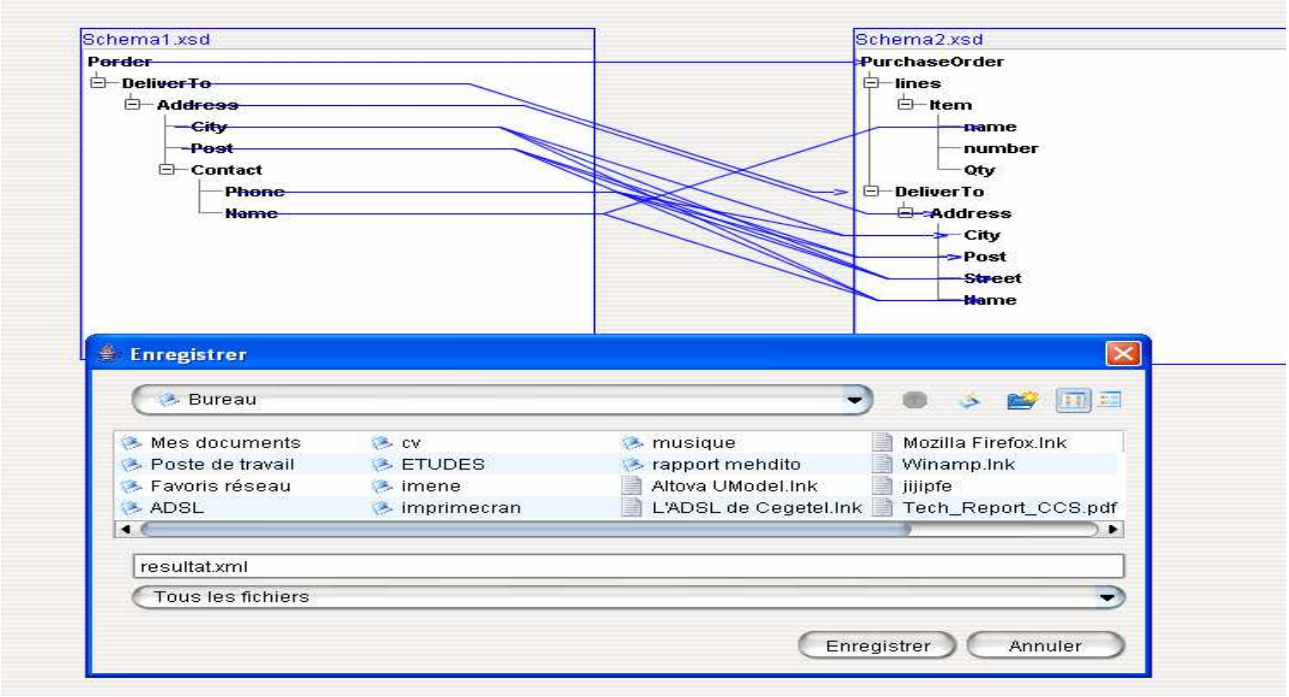


Figure 25 Sauvegarde du résultat du Matching

3.2.6 Choix des fonctions de Mapping

Pour choisir la fonction de Mapping, il suffit de cliquer avec le bouton droit de la souris sur un ligne. Un menu représentant un choix entre les fonctions XME et des conditions s’affiche. Quand on clique sur “XME Functions”, un box s’affiche sur l’interface sur le lien sélectionné (figure 26). On peut dans ce cas comme pour l’exemple présenté dans la figure , choisir la fonction concat entre LastName et FirstName et on aura comme élément cible généré Name. L’utilisateur a libre choix de ces fonctions de mappage ce qui représente une facilité d’utilisation (figure 27).

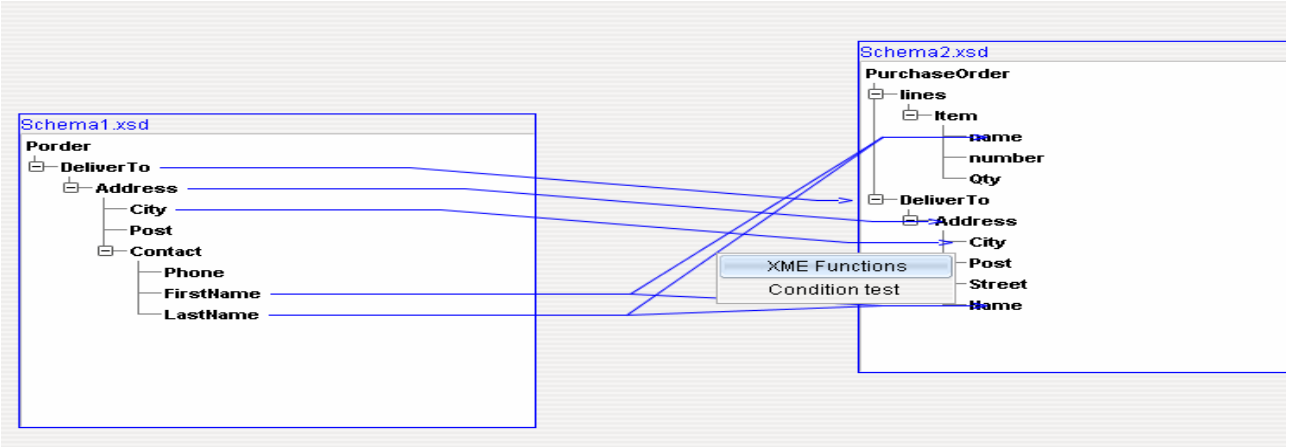


Figure 26 Choix des fonctions de Mapping

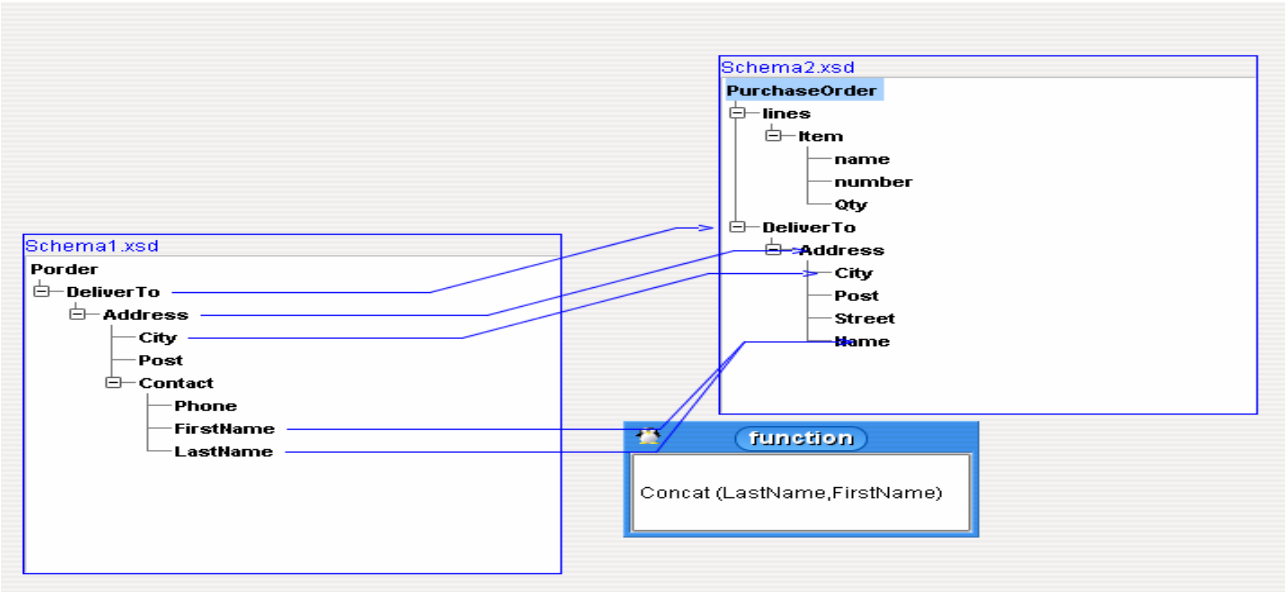


Figure 27 Mapping entre les schémas

4. Difficultés techniques

On a rencontré plusieurs problèmes lors de la réalisation de ce prototype qu'on a pu soulever :

- La difficulté à intégrer EXSMAL dans une nouvelle interface.
- Difficulté à réaliser une interface où l'on peut faire des drag et drop des objets manipulés sur l'interface, le fait de pouvoir associer des lignes et les faire bouger avec les schémas.
- La réalisation correcte du matching entre les schémas.
- Le choix des bibliothèques et des jars.

5. Conclusion

Dans ce chapitre, nous avons présenté le prototype réalisé. Ce prototype a pour avantage par rapport à d'autres outils existants de réaliser sur la même interface le Matching et le Mapping des schémas. De plus, il représente une vraie convivialité d'utilisation du fait de pouvoir choisir plusieurs schémas sources afin de réaliser des Matchings de type (1-1,1-n,n-1,n-m) , de pouvoir déplacer ces schémas sur l'interface, d'ajouter des fonctions de Mapping et de générer le résultat en XQuery.

Conclusions et Perspectives

L'intégration de sources de données hétérogènes dans le contexte du Web est un problème d'actualité. L'apport du langage XML comme standard d'échange et de représentation de ces données est indéniable et facilite la mise en place de différentes approches. En effet, XML permet de représenter des données indépendamment de leurs présentations et de leurs stockages. Aussi, tous les systèmes d'intégration actuels par médiateur utilisent XML comme modèle pivot. Cependant, XML ne permet pas d'éviter les problèmes de conflits d'hétérogénéité sémantique et structurelle. Ces problèmes peuvent être résolus en utilisant les schémas XML et en se basant sur les techniques de Matching et de Mapping de ces schémas.

Ayant établi une étude sur les approches de Matching, de modèles de découverte d'expression de mappage et des outils de Mapping, nous avons proposé une architecture ASMADE (Automated Schema Matching For Documents Exchange). L'algorithme EXSMAL y est intégré pour réaliser le Matching entre deux schémas source et cible. On applique un filtre afin de raffiner les résultats de maching et avoir des correspondances plus précises. Ensuite, le résultat sera représenté selon le modèle XME (XML Mapping Expression) en appliquant de plus un ensemble d'opérateurs de transformations afin de réaliser le Mapping et trouver la vraie relation sémantique entre les schémas. Une fois les Mappings sémantiques définis, le générateur XQuery ASMADE sera utilisé pour réaliser la transformation de documents XML basés sur un schéma XML en d'autres documents basés sur les schémas XML. Nous avons développé un prototype qui met en place cette architecture.

Néanmoins, ce prototype n'est pas celui qui contient parfaitement toutes les fonctionnalités qu'un outil réalisant le Matching et le Mapping devrait avoir. Il est d'ailleurs en cours de développement et pourrait être amélioré selon plusieurs points de vue :

- Intégrer en totalité les opérateurs de transformation définis.
- Revoir des techniques de traitement linguistique plus sophistiques et les intégrer.
- Le résultat de ce travail devrait s'appliquer aussi bien aux schémas de BD que les schémas XML, les DTD, etc.

Ce projet fait partie d'un effort continu de recherche impliquant différents groupes pour espérer fournir, dans un long terme, une plateforme intégrée permettant la gestion de schémas, d'ontologies mais aussi de matching des taxonomies, d'arbres de vie (Tree of life).

Dans un autre registre, ce stage m'a permis d'élargir et approfondir mes connaissances sur les systèmes d'information, d'intégration de données et m'a rendu plus ambitieuse et plus motivée pour continuer dans ce travail de recherche.

Bibliographie

Ouvrage ou Mémoire:

[1] Rami Rifaieh. « Utilisation des ontologies contextuelles pour le partage sémantique entre les systèmes d'information dans l'entreprise ». Thèse, Ecole doctorale: Informatique et Information pour la société (EDIIS-EDA 335), Institut National des Sciences Appliquées de Lyon.

[2] Khaled Herzi, « Adaptation et Raffinement des correspondances dans EXSMAL pour la transformaton des documents XML », Mémoire de Master encadré par Nabila BENHARKAT, juin 2005.

[3] MAZLOUT Haïkel « Rapport d'Apprentissage Master2 MIAGE HyperObjects », Grenoble2005.

[4] Uddam CHUCKMOL « Mappage entre les messages EDI/XML », Mémoire de DEA, Septembre2004.

Articles de revue:

[5] Dr. Edward A. Fox, Divya Rangarajan, SchemaMapper: « A tool for visualization of schema Mapping Independent Study Report», 2004.

[6] Hong-Hai Do and E.Rahm. « COMA-A system for flexible combination of Schema Matching approaches». In Proceeding of the 28th VLDB Conference, August, 2002, Hong Kong, China.

[7] Rami Rifaieh, Uddam Chukmol and Nabila Benharkat. «EXSMAL:EDI/XML semi-automatic Schema Matching Algorithm » In IEEE CEC 2005, July 19-22,2005, Technische Universität München, Germany.

[8] A.Boukottaya, C.Vanoirbeek, F.Paganelli, O.Abou Khaled. « Automating XML Transformations: A conceptual modelling based approach ». Media Research Group, EPFL(Swiss Federal Institute of Technology) 1015 Lausanne, Switzerland, 2004.

[9] Zoubida Kedad, Xiaohui Xue, « An automatic tool for discovering complex Mappings », Laboratoire PriSM, Université de Versailles, France, 2005.

[10] David W.Embley, Li Xu, Yihong Ding, « Automatic Direct and Indirect Schema Mapping: Experiences and Lessons Learned », SIGMOD Record, Vol,33.No.4,December 2004.

[11] Renée J.Miller, Lara M.Haas, Mauricio A.Hernandez. « Schema Mapping as Query Discovery », Proceeding of the 26th VLDB Conference, Cairo, Egypt, 2000.

[12] R.J Miller et al. « The Clio project: managing the heterogeinity ». SIGMOD Record, 2001.

[13] Philip A. Bernstein. « Applying Model Management to Classical Meta Data Problems », Microsoft Research One Microsoft Way, 2003.

[14] Huiyong Xiao, Isabel F. Cruz, Feihong Hsu. « Semantic Mapping for the Integration of XML and RDF Sources »,Department of Computer Science, University of Illinois at Chicago, USA, 2004.

[15] J.Madhavan, P. A. Bernstein and Rahm. « Generic Schema Matching with Cupid». Proceedings of the 27th VLDB Conference, 2001, Rome, Italy. pp.49-58.

[16] S.Melnik, H.Garcia-Molina and E. Rahm. « Similarity Flooding: A versatile Graph Matching approaches». In Proceeding (ICDE), 2002, San Jose, California, USA.

[17] S.Melnik, E.Rahm and P.A. Berstein. «Rondo: A programming Platform for Generic Model Management. » In Proceedings of SIGMOD Conference, June 9-12, 2003, San Diego, California, USA. pp. 193-204.

[18] **Mong Li Lee, Liang Huai Yang, Wynne Hsu, Xia Yang**. « XClust : Clustering XML Schemas for Effective Integration ». School of Computing, National University of Singapore 3 Science Drive2.

[19] L. Kurgan, W. Swiercz and K. J. Cios. «Semantic Mapping of XML tags using inductive machine learning». In Proceedings of the 2002 International Conference on Machine Learning and Application (ICMLA'02), Las Vegas, Nevada, USA. pp. 99-109

[20] J. Madhavan, P. A. Bernstein et al. «Corpus-based Schema Matching». In Proceedings of the 18th International Joint Conference on Artificial Intelligent (IJCAI'03), 2003, Acapulco, Mexico. pp. 49 - 53.

[21] S.Castano, A.Ferrara and S.Montelli. «H-MATCH: An Algorithm for Dynamically Matching Ontologies in Peerbased Systems ». In Proceedings of the SWDB 2003 Conference, September, 2003, Berlin, Germany.pp. 231- 250.

[22] A.H Doan, J.Madhavan, P.Domingos and A.Halevy. «Learning to map between Ontologies On the Semantic Web». In Proceeding of the 11th International Conference on Word Wide Web, May 7-11, 2002, Honolulu, Hawaii, USA.pp.662-673

[23] Aumüller, D., Do, H.H., Massmann, S., Rahm, E «Schema and Ontology Matching with COMA++».Proc. SIGMOD 2005 (Software Demonstration), Baltimore, June 2005

[24] Amar ZERDAZI, Myriam LAMOLLE. « HyperSchema XML: Un modèle d'intégration par enrichissement sémantique de schémas XML ».MajecSTIC 2005 .

[25] Khaled Herzi, Aïcha-Nabila Benharkat, Youssef Amghar « Refinement of correspondences

in EXSMAL for XML Document transformation" to appear in WBC'06 in the 17th International Conference on Database and Expert System Applications DEXA'06.

[26] George H. L. Fletcher, Catharine M. Wyss, « Discovering Complex Mapping Expressions with the TUPELO Data Mapping System », *IHIS'05*, November 4, 2005, Bremen, Germany.

Adresses Web:

[27] User and Reference Manual, **Altova MapForce** 2005, http://www.altova.com/, United States, 2005.

[28] Stylus Studio Product, http:// Stylusstudio.com

[29] Clio XML Demo, http://www.almaden.ibm.com/cs/clio/xmlDemoP1.html

[30] BEA Weblogic Workshop 8.1 (beta) JumpStart Guide, BEA Systems, http://www.bea.com, Mars 2003.

[31] *Consortium W3C*, http://ww.w3.org/XML

[32] Visual XSLT, http://aspn.activestate.com/ASPN/Downloads/VisualXSLT

[33] TIBCO XML Transform, http://www.tibco.com/software/metadata/xmltransform.jsp

[34] Redix AnyToAny XML GUI Mapper, http://www.redix.com/dtd13.htm

[35] Cape Clear, http://www.capeclear.com/

[36] Adeptia XML Mapper, http://www.adeptia.com

[37] XQuery, http:// www.xquery.com

Annexe A: Schéma1 et Schéma2

Schéma1:

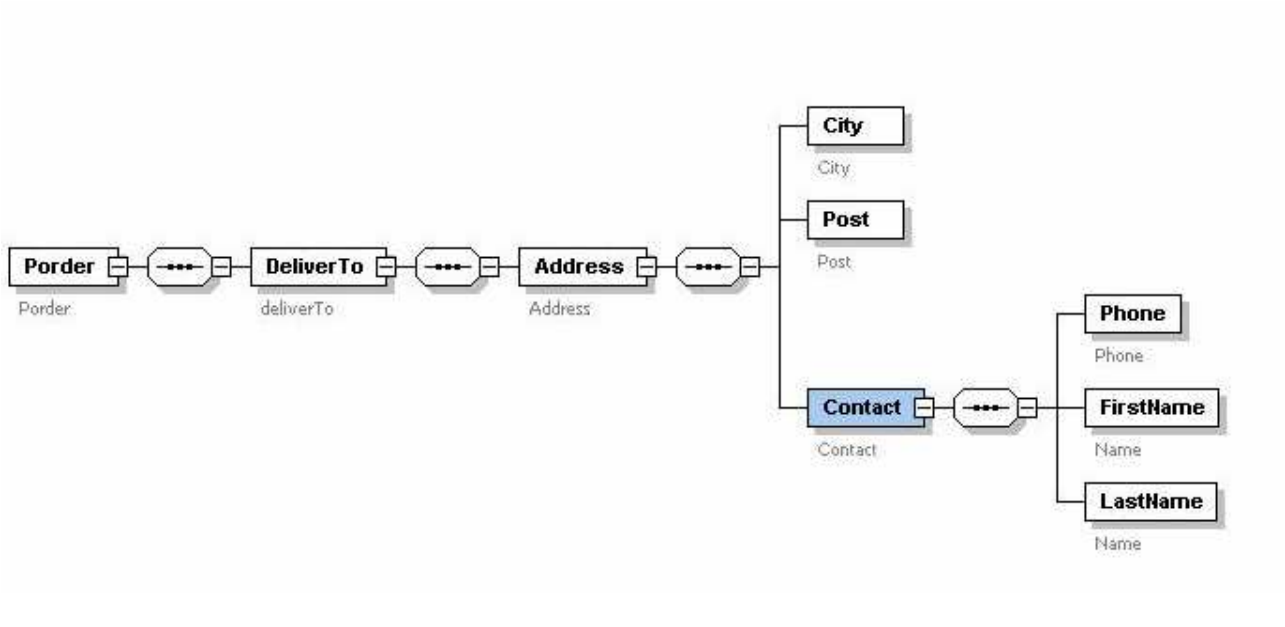
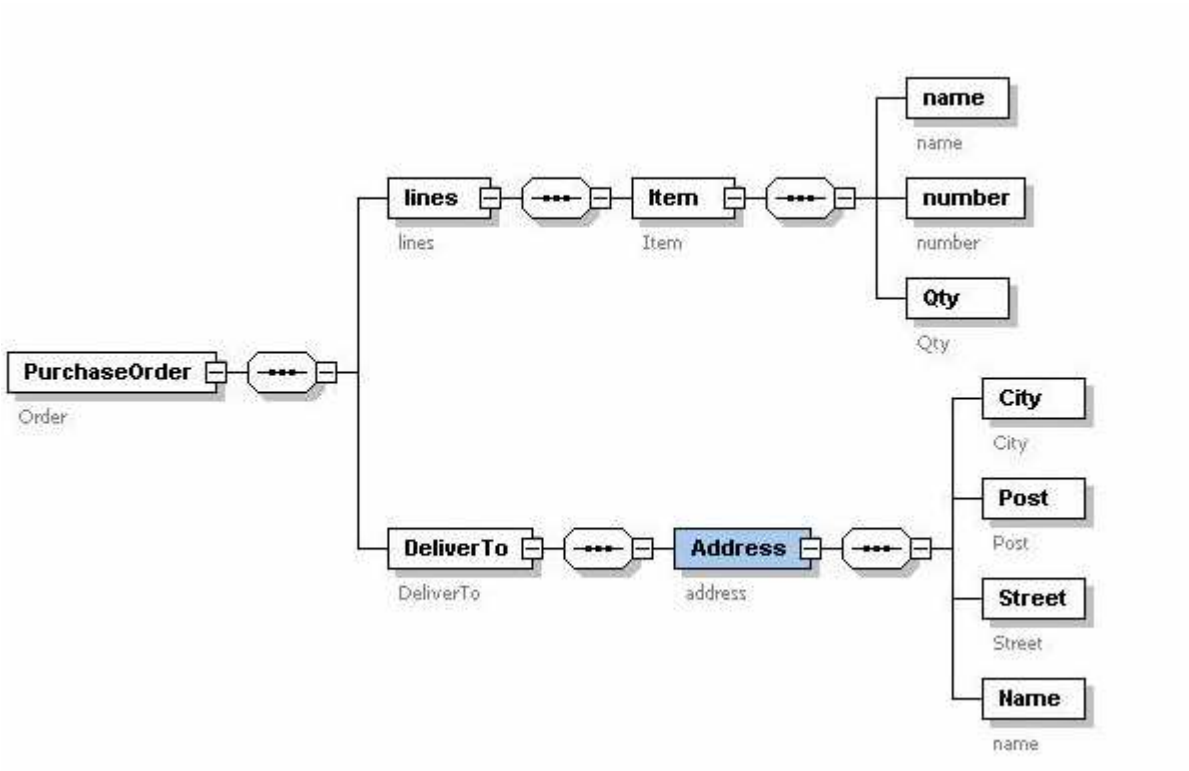


Schéma2:



Annexe B: Requête de transformation de schémas

Transformation du Schema1.xsd vers Schema2.xsd

On suppose qu'il existe une instance de Schema1.xsd intitulée Schema1.xml. La requête XQuery qui permet de générer un nouveau fichier Schema2.xml à partir du fichier source et en respectant les Mappings entre les deux schémas est la suivante :

```
define function fomratter($fichier_donnees)
{
  for $donnees in $fichier_donnees
        return
        (
                <PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="C:\Eclipse 3.0\workspace\asmade2\schema\Schema2.xsd">
                        <lines>
                                <Item>
                                        <name/>
                                        <number/>
                                        <Qty/>
                                </Item>
                        </lines>
                        <DeliverTo>
                                <Address>
                                        <City> $donnees/Porder/DeliverTo/Address/City <City/>
                                        <Post>$donnees/Porder/DeliverTo/Address/Post<Post/>
                                        <Street/>
                                        <Name>
                                       concat($donnees/Porder/DeliverTo/Address/Contact/FirstName,
                                              $donnees/Porder/DeliverTo/Address/Contact/LastName) <Name/>
                                </Address>
                        </DeliverTo>
                </PurchaseOrder>

        )
}


let $fichier :=document (schema1.xml)
retrun
(
writeInto(formatter($fichier), schema2.xml)

)
```

ملخّص:

إنّ تصميم نظم قواعد البيانات قد بات شيئاً فشيئاً متعدّد المصادر. هاته المصادر متغايرة الخواصّ، متباعده، متواجده على الإنترنات أو من خلال شبكات المؤسّسات. وقد بدت معالجة المسائل المطروحة بسبب تغاير خواصّ البيانات واحدة من أهمّ صعوبات هذا النّوع من التّصميم. ولهذا، فقد وجب تصنيف هذه الصّعوبات ضمن ثلاث وهي : إدماج الرّسوم البيانيّة، توليد عبارات التّخطيط، وتحويل البيانات.

من مشمولات إدماج الرّسوم البيانيّة حلّ مشاكل عدّة نذكر منها إختلاف العبارات المستعملة في مختلف لغات البرمجيّة اللّتي صيغت بها متعدّد مصادر البيانات، تعدّديّة أوجه النّظر تجاه العالم الواقع، عدم تطابق شروط التّكامل مع أحكام المعاملات، تعدّد النّماذج التّمثيليّة، إلخ...

ينتج عن إدماج الرّسوم البيانيّة الحصول على مخطّط إجمالي موحّد، أو إنتاج جملة من أحكام التّواصل بين المخططات.

وقد يتسبّب إختلاف خصائص مصادر البيانات في عدة مسائل، لذا فقد تمّ إقتراح جملة من أنظمة العدّ بهدف مقارنة مخططات XML **و إيجاد درجات تشابه بين العناصر.**

وفي هدا النّطاق، فقد تمّ الإتّفاق على جملة من عبارات التّخطيط و صمّمت مجموعة من الأدوات لتوليد المخططات بين رسوم XML **وكذلك أصناف أخرى للبيانات.**

المطابقة الآليّة للرّسوم لتبادل البيانات) اللّذي يجمع كلّ هاته الخاصيّات بالقيام بعمليّات المطابقة، تخطيط) ASMADE **يطرح هذا العمل تعريفاً و إقتراحاً لتصميم** XQuer **والسّماح بتحويل أصماف الملفات بفضل مولّد ،**XME **و من ثمّ تحويلها نحو صنف** XML **رسوم**

Conception et Réalisation d'un outil de génération automatique de Mappage pour la transformation de documents XML

Résumé :

La conception de systèmes d'information est de plus en plus multi sources. Ces sources de données sont hétérogènes, distribuées, accessibles par le Web ou par un réseau d'entreprises. La résolution des problèmes posés par l'hétérogénéité est une des principales difficultés dans cette conception. On peut classer ces problèmes en trois catégories : intégration des schémas, la génération d'expressions de mappage et la transformation de données.

L'intégration des schémas doit résoudre les problèmes des terminologies utilisées dans les différentes sources, de multiplicité des vues du monde réel, de la non-conformité des contraintes d'intégrité et des règles de gestion, de la multiplicité des modèles de représentation, etc. L'intégration de schémas peut soit produire un schéma global unique, soit produire un ensemble de règles de correspondances entre les schémas. Du fait de l'hétérogénéité des sources, de nombreux problèmes peuvent subsister. Plusieurs algorithmes ont été proposés pour comparer les schémas XML et établir les degrés de similarité entre les éléments. Plusieurs approches ont été définies pour traiter les expressions de mappages et différents outils ont été conçus générant le Mapping entre les schémas XML et d'autres formats de données.

Dans ce travail, nous avons proposé et défini une architecture ASMADE (Automated Schema Matching For Documents Exchange) qui réunit toutes ces spécificités en réalisant le Matching, le Mapping des schémas XML ensuite leur représentation en XME (XML Mapping Expression) ; la génération en XQuery pour permettre la transformation de documents

Mots clés :

Schéma XML, Interopérabilité, Matching des schémas, Mapping des schémas, modèle d'expression de mappage, Transformation

Conception and Realisation of an automatic Mapping generation tool for XML documents transformation

Abstract:

Information systems design is increasingly multi-source. These data sources are heterogeneous, distributed and accessible by the Web or by an enterprise network. The heterogeneity problem is the principal difficulty in this design. In our context, we can classify these problems in three categories: schema integration, generating Mapping expression and data transformation. The schema integration has to solve the terminology problems that are used in different sources, the multiplicity of views in the real world, the nonconformity of the integrity constraints and management rules, the multiple representation models, etc. The schema integration can either produce a unique global schema or produce a group of corresponding rules between the schemas. In all these contexts, sources heterogeneity araises many problems. For this purpose, several algorithms have been proposed to compare the XML schemas and to establish similarity degrees between the elements. Many approaches have been defined to treat the Mapping expressions and different tools have been proposed for generating the Mapping between XML schemas and other data formats. In our work, we proposed and defined architecture ASMADE (Automated Schema Matching For Documents Exchange) for automated transformation of schema instances. The ASMADE Framework provides a general outline to perform Matching, Mapping and transformations with minimal and limited technical skills.

Key Words:

XML schemas, interoperability, Schema Matching, Schema Mapping, Mapping Expression Model, transformation.

Intitule et adresse complète du laboratoire :

Laboratoire: LIRIS (Laboratoire d'Informatiques en Images et Systèmes d'Information)
Adresse: INSA de Lyon Bâtiment Blaise PASCAL 7, avenue Jean Capelle 69621 Villeurbane Cedex
Tél.: +33 4 72 43 60 55
Fax: +33 4 72 43 60 71
Site : http://www.insa-lyon.fr