Un tour d'horizon des approches pour la manipulation des données du web

Arnaud Blouin, Olivier Beaudoux, Stéphane Loiseau

Un tour d'horizon des approches pour la manipulation des données du web

Arnaud Blouin[*], Olivier Beaudoux[*] et Stéphane Loiseau[**]

[*] GRI, Groupe ESEO. 4, rue Merlet de la Boulaye, 49009 Angers, France
{arnaud.blouin, olivier.beaudoux}@eseo.fr

[**] LERIA, Université d'Angers. 2, Bd Lavoisier 490045 Angers, France
stephane.loiseau@univ-angers.fr

RÉSUMÉ. Le passage au web 2.0 a renforcé le principe selon lequel les données doivent être séparées de leurs présentations. En même temps, le nombre des formats de documents XML a largement évolué sur le web. Les bases de données sont également utilisées pour le fonctionnement des sites. Ces constats impliquent la mise en œuvre d'approches et d'outils associés dédiés à la manipulation des données du web. Certaines de ces approches travaillent directement sur les données, tandis que d'autres se basent sur les schémas de ces mêmes données. L'Ingénierie Dirigée par les Modèles (IDM) forme également une approche candidate à la manipulation des données du web. Dans cet article, nous présentons et comparons les différentes techniques majeures de manipulation de données afin de mettre en exergue leurs avantages et leurs inconvénients pour le problème des données du web.

ABSTRACT. The transition to the web 2.0 has reinforced the principle which states that data should be separated from their presentations. In the same time, the number of XML document formats on the web has greatly evolved. Databases are also often used for the functioning of websites. These findings imply the need to implement tools and approaches dedicated to the manipulation of data on the web. Some of these approaches work directly on the data, while others are based upon the schema of these data. The Model-Driven Engineering (MDE) also allows the manipulation of web data. In this paper, we compare and contrast the different major techniques for data manipulation in order to highlight the pros and cons of these methods for the problem of web data.

MOTS-CLÉS. données du web, transformation de données, translation de données, XML

KEYWORDS. web data, data transformation, data translation, XML

1 Introduction

Depuis le web 2.0, les données du web sont devenues réellement omniprésentes sur Internet et se divisent en deux catégories : les bases de données et les documents XML (Abiteboul *et al.*, 2000). Les bases de données sont utilisées pour la gestion de sites de e-commerce et d'applications du web 2.0 (Atzeni *et al.*, 1997), tels que les wikis (p. ex. *Wikipedia*), les gestionnaires de projets (*Trac*) et les webmails (*GMail*). Par ailleurs, les documents XML permettent la gestion des flux d'informations (RSS), la description des interfaces des RIA (*Rich Internet Application*) et la définition des pages web (XHTML).

Au cœur de ces systèmes, la manipulation de données peut être divisée en deux domaines d'application. Tout d'abord, elle permet l'interopérabilité entre des bases de données et par conséquent entre les systèmes qui les utilisent. Ce principe est appelé la translation de schémas puisque le but d'une translation est d'établir un pont entre deux ensembles de données homogènes exprimant un même concept. De plus, la manipulation de données permet la création de présentations. Ce principe est appelé la transformation de schémas étant donné que les schémas sources et le schéma cible n'expriment pas le même concept. Par exemple, la transformation s'applique à la construction d'une page web à partir d'une base de données.

Actuellement, la manipulation de données du web (MDW) peut être divisée en trois groupes. Le premier rassemble les techniques et les langages manipulant directement les données, tels qu'XSLT et XQuery. Le principal inconvénient de cette approche est qu'elle ne vérifie pas la structure des données, ce qui rend la manipulation sujette à erreur. Pour éviter ces inconvénients, le deuxième groupe concerne les techniques travaillant au niveau des schémas ; au lieu de définir une transformation entre des données, ces techniques travaillent au niveau de leur schéma. Le troisième groupe comprend les approches dédiées à la transformation dans le cadre d'IDM (*ingénierie dirigée par les modèles*). Même si l'IDM et ses langages associés (ATL, QVT) visent à répondre à un problème plus général que celui de la MDW, une transformation de modèles peut être développée dans ce but. Dans ce contexte, une source de données est considérée comme un modèle et son schéma comme un métamodèle.

Cet article étudie les différentes approches permettant la MDW selon les trois groupes présentés précédemment, afin d'identifier le pour et le contre de chacune d'entre elles pour les problèmes de translation et de transformation de schémas.

La section suivante présente les différents domaines d'application de la MDW. La section 2.2 propose une classification des différentes approches. Les sections 3, 4 et 5 présentent les méthodes et les langages des trois groupes définis dans la section 2.2. Le tableau 1 de l'annexe, récapitule les caractéristiques des approches majeures présentées pour le problème de la MDW.

2 Applications et classification

Cette section met en exergue, dans un premier temps, l'importance de la MDW en présentant ses domaines d'application majeurs. Dans un second temps, elle présente une classification des différentes approches dédiées au problème de la MDW.

2.1 Domaines d'application

Dans le web 1.0, les données consistaient en des pages HTML, écrites le plus souvent à la main, et des bases de données (Florescu *et al.*, 1998). Dans ce contexte, les deux principaux besoins de la MDW étaient la communication entre les bases de données et la création de vues et de formulaires permettant la visualisation (Abiteboul, 1999; Abiteboul *et al.*, 1999) et l'édition (Nguyen *et al.*, 1996) de ces données. Ces besoins relèvent des deux domaines de manipulation de données présentés par la figure 1 : la translation et la transformation de schémas. Le passage vers le web 2.0 a renforcé ces besoins en confortant le principe de la séparation entre les données et leurs présentations. La manipulation de données s'est, de plus, étendue aux documents depuis l'avènement d'XML (W3C, 2006a). Les bases de données sont également largement utilisées, notamment pour le fonctionnement des RIA.

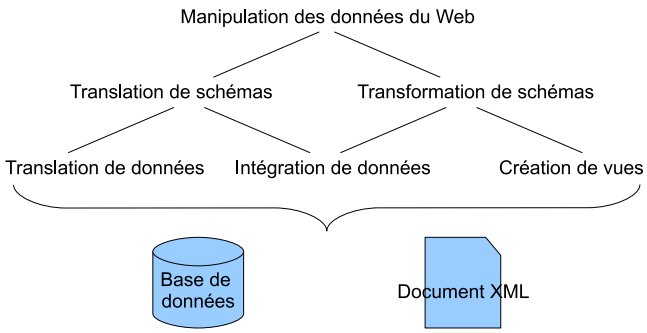
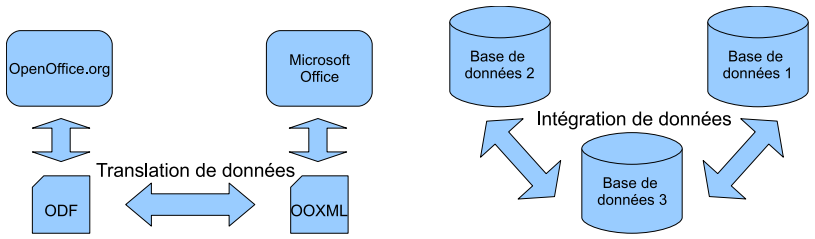


Figure 1: Différents domaines d'application de la manipulation de données du web

Une manipulation de données est dite *translation de schémas* lorsque la sémantique du ou des schémas sources est la même que celle du schéma cible. La translation de schémas permet l'interopérabilité entre des données exprimant un même concept mais de manière différente. Il est courant, en effet, que des données hétérogènes soient représentées dans différents formalismes (p. ex. schéma relationnel, DTD ou XML Schema) ou dans un formalisme identique mais de manière distincte. La translation de schémas a pour but l'homogénéisation d'un ensemble de données sources hétérogènes. Elle peut être divisée en deux sous-domaines majeurs : la translation de données et l'intégration de données. La *translation de données*, également appelée l'échange de données, est le problème visant à faire passer des données structurées d'un schéma source vers un schéma cible (Fagin *et al.*, 2003; Abiteboul *et al.*, 2002; Atzeni, 2006; Kolaitis, 2005). La figure 2(a) présente un exemple de translation de données où le but est de créer un pont entre un document au format ODF (*Open Document Format*) et un autre au format OOXML (*Office Open XML*) qui définissent le même concept (la notion de document bureautique) différemment.

L'intégration de données vise à combiner des données résidant dans différentes sources afin de fournir à l'utilisateur une vue unifiée et globale de ces données (Batini *et al.*, 1986; Lenzerini, 2002). La figure 2(b) illustre le principe de l'intégration de



(a) La translation de données (b) L'intégration de données

Figure 2: Principe de la translation et de l'intégration de données

données avec un exemple : les sources contiennent les données d'origine tandis que la cible ne fournit qu'une vue virtuelle regroupant tout ou partie des données sources.

Bien qu'à l'origine la translation de schémas concernait essentiellement le domaine des bases de données, elle s'est étendue au domaine des documents comme l'illustre la figure 2(a). Ce problème s'est accentué depuis l'avènement de l'XML et du web 2.0 où de nombreux formats de documents exprimés en XML sont apparus. L'ODF d'OASIS et l'OOXML de Microsoft pour les documents bureautiques, ou encore les formats RSS et Atom pour la syndication de flux, élément essentiel du web 2.0, illustrent bien ce constat.

Une manipulation de données est dite *transformation de schémas* lorsque la sémantique du ou des schémas sources diffère de celle du schéma cible. La transformation de schémas peut être divisée en deux sous-domaines majeurs : l'intégration de données et la création de vues. *L'intégration de données*, présentée précédemment en tant que translation de données, peut également être considérée comme de la transformation de schémas dans certains cas. En effet, si les schémas sources et le schéma cible ont la même sémantique, alors il s'agit d'une translation de schémas. Par exemple, la fusion de deux schémas définissant chacun le concept d'individu, en un schéma global ayant le même but, est un problème de translation. Par contre, la fusion d'un schéma spécifiant le concept d'individu avec un autre schéma définissant le concept d'historique de navigation web, en un schéma global dédié à l'étude statistique de la navigation web des individus, est considérée comme de la transformation de schémas. Dans certains cas la classification d'une intégration de données peut être subtile et dépend essentiellement du contexte et de la sémantique des schémas concernés. L'autre sous-domaine majeur de la transformation de schémas est la *création de vues*. Abiteboul (1999) présente les vues comme des outils permettant à des utilisateurs de voir des données de différents points de vues. Les exemples de créations de vues sont nombreux et indispensables au fonctionnement du web 2.0. Les blogs sont, notamment, l'assemblage d'une base de données contenant les messages postés et leurs réponses, et d'une présentation généralement synthétisée par une page web.

2.2 Classification des approches de manipulations des données du web

La figure 3 synthétise notre classification des différents grands groupes d'approches de la MDW qui ressortent de la littérature.

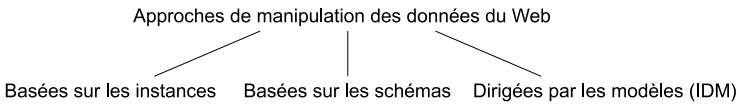


Figure 3: Classification des approches de la manipulation des données du web

Les approches *basées sur les instances*, c'est-à-dire sur les données elles-mêmes et non sur leur schéma, s'appliquent principalement à la manipulation de documents structurés. Cette tendance s'est accentuée depuis le consensus établissant XML comme le langage standard pour le stockage et l'organisation des données et de ses technologies, notamment XSLT et XQuery.

Ensuite, les approches *basées sur les schémas* considèrent le schéma comme l'élément central de la manipulation et non ses données instances. Ces approches s'appliquent aux bases de données, dont le schéma est principalement défini en schéma relationnel, mais également aux documents structurés (schéma XML).

Troisièmement, les approches de *l'ingénierie dirigée par les modèles* (IDM) se rapprochent des approches basées sur les schémas dans le sens où les données ne sont plus considérées comme l'élément central d'une manipulation mais comme une instance possible de son schéma.

Dans les trois sections suivantes, nous détaillons les groupes exposés dans la figure 3 notamment en présentant les différents langages et cadres de travail existants.

3 Approches basées sur les instances

Pour illustrer les approches majeures présentées dans les sections 3, 4 et 5, un même exemple de manipulation de données est utilisé. La source et la cible de cette manipulation sont respectivement le document XML de la figure 4(a) et le document SVG de la figure 4(b). Cet exemple vise à créer une présentation graphique en SVG (W3C, 2003) à partir de données XML représentant un blog. Dans un soucis de simplicité, les commentaires pouvant être attachés aux billets d'un blog ont été omis. Le choix du format SVG, au détriment d'autres formats web plus courant comme XHTML, est dû au fait que l'utilisation de SVG permet de mettre en évidence certains points importants des approches présentées.

Les approches basées sur les instances travaillent directement sur des données. L'avantage de ce principe est de permettre le développement de manipulations sans avoir recours à la définition du schéma des données qui dans certains cas n'existe pas ou n'est pas disponible. Ce principe peut s'avérer être un inconvénient lors de manipulations de données dont le schéma est complexe, puisque les données cibles peuvent

(a) Un fichier XML

```
<blog nom="mon Blog">
 <billets>
  <billet num="1">
   <titre>Mon premier
     billet</titre>
   <contenu>Ceci est
   le contenu de mon
   premier billet.
   </contenu>
   <date>11-17-2008</date>
   <auteur>Arnaud B.</auteur>
  </billet>
 </billets>
</blog>
```

(b) Exemple d'une présentation SVG

```
<s:svg xmlns:s="http://www.w3.org/2000/svg">
  <s:g font-size="24" fill="#000000">
    <s:text x="-280" y="20" text-anchor="middle"
      transform="rotate(-90)">mon Blog</s:text>
    <s:g id="1">
      <s:rect y="30" x="40" fill="#e6e6e6"
        stroke="#000000" height="180" width="630"/>
      <s:text font-weight="bold" y="60" x="60">Mon
        premier billet</s:text>
      <s:text y="125" x="55">Ceci est le contenu
        de mon premier billet.</s:text>
      <s:text font-style="italic" text-anchor="end"
     y="195" x="620">Arnaud B. - 11-17-2008</s:text>
    </s:g>
  </s:g>
</s:svg>
```

Figure 4: Exemple du blog

ne pas être conformes à leur schéma.

Etant donné que chaque approche recensée utilise un langage de programmation, nous les avons classées en fonction de leur type : les langages déclaratifs et de requêtes. Il existe également des approches qui étendent des langages de programmation courants afin de faciliter la manipulation de données XML, comme XJ (Harren *et al.*, 2005) et XACT (Kirkegaard *et al.*, 2004) pour Java, XTATIC pour C# (Gapeyev *et al.*, 2006), HaXml pour Haskell (Wallace *et al.*, 1999) ou XCentric pour Prolog (Coelho *et al.*, 2007). De plus, des méthodes graphiques permettent la définition de manipulations comme par exemple, VXT qui génère des programmes Circus et XSLT (Pietriga *et al.*, 2001), visXcerpt générant des programmes Xcerpt (Berger *et al.*, 2003), ou encore l'éditeur commercial Stylus Studio[1] générant des programmes XSLT et XQuery.

3.1 Les langages déclaratifs

La programmation déclarative s'applique de manière homogène à la manipulation de données. En effet, un programme déclaratif se compose d'un ensemble de règles décrivant quelles données doivent être manipulées, les opérations à réaliser et les données cibles, sans stipuler comment la manipulation doit être effectuée.

Le langage déclaratif le plus connu est certainement XSLT (W3C, 2007b; Kay, 2004b; Drix, 2002). Ce langage permet la transformation de documents XML en d'autres documents textuels, XML ou non. Le principe général de la programmation en XSLT consiste à définir des règles de transformation (*templates*) à appliquer au document source XML. La navigation dans le document source XML s'effectue de manière arborescente grâce au langage XPath (W3C, 2007a; Kay, 2004a). L'exemple suivant présente une transformation XSLT de l'exemple de la figure 4.

Le programme XSLT de la figure 5 se compose de deux règles : la première, ligne 4, définit les actions à réaliser, en l'occurrence la création du corps du document SVG

[1] http://www.stylusstudio.com/

```
1 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
2   version="1.0" xmlns:s="http://www.w3.org/2000/svg">
3   <xsl:output method="xml" encoding="utf-8" indent="yes"/>
4     <xsl:template match="blog">
5       <s:svg>
6         <s:g font-size="24" fill="#000000">
7         <s:text x="{-count(billets/billet)*140}" y="20" text-anchor="middle"
8           transform="rotate(-90)"><xsl:value-of select="@nom" /></s:text>
9           <xsl:apply-templates select="billets"/>
10        </s:g>
11      </s:svg>
12    </xsl:template>
13  <xsl:template match="billet">
14    <s:g id="{@num}">
15      <xsl:variable name="y"><xsl:value-of select="(position()-2)*130"/>
16      </xsl:variable>
17      <s:rect y="{$y+30}" x="40" fill="#e6e6e6" stroke="#000000" height="180"
18        width="630"/>
19      <s:text font-weight="bold" y="{$y+60}" x="60"><xsl:value-of select="titre"/>
20      </s:text>
21      <s:text y="{$y+125}" x="55"><xsl:value-of select="contenu"/></s:text>
22      <s:text font-style="italic" y="{$y+195}" x="620" text-anchor="end">
23        <xsl:value-of select="concat(concat(auteur, ' - '), date)"/></s:text>
24    </s:g>
25  </xsl:template> <xsl:template match='text()|@*' />
26 </xsl:stylesheet>
```

Figure 5: Transformation du blog en XSLT

cible, lorsque le nœud courant a pour nom `blog`. Cette règle en appelle une seconde, ligne 13, qui construit un bloc d'objets SVG pour chaque billet.

3.2 Les langages de requêtes

Les langages de requêtes du web ont pour but de sélectionner, *via* des requêtes, des données dans des bases de données et, plus récemment, dans des documents structurés.

XQuery (W3C, 2006b) est un langage de requêtes pour documents XML dont il est souvent dit qu'il est pour XML, ce qu'est SQL pour les bases de données. Si XQuery et XSLT présentent des similarités en termes de fonctionnalité (ils analysent tous les deux des sources XML, et XQuery peut être considéré comme un langage de transformation), leur principale différence est culturelle ; le premier est orienté pour les bases de données alors que le second l'est pour le document. Les deux langages font référence à XPath ; XSLT l'utilise de manière indépendante afin de parcourir un document XML, alors qu'XQuery peut être considéré comme une surcouche d'XPath.

La figure 6 définit un programme XQuery reprennant le problème de la transformation du blog en un document SVG. Ce programme interroge le document XML source, spécifié à la ligne 2, et retourne les données cibles. Pour chaque billet du blog, l'itération à la ligne 9 lance une requête pour construire un bloc d'objets SVG.

Une des particularités d'XQuery est que ce langage peut être considéré comme un langage de transformation étant donné que les données cibles peuvent être restructurées, contrairement à des langages de requêtes classiques comme SQL. XQuery peut donc être considéré comme un langage de manipulation de données réalisant aussi bien de la transformation que de la translation de schémas.

```
declare namespace svg="http://www.w3.org/2000/svg";
let $doc := doc("blog.xml")
return
  <s:svg xmlns:s="http://www.w3.org/2000/svg">
    <s:g font-size="24" fill="#000000">
     <s:text x="{-count($doc/blog/billets/billet)*140}" y="20"
      text-anchor="middle" transform="rotate(-90)">
      {data($doc/blog/@nom)}</s:text> {
     for $i in 1 to count($doc/blog/billets/billet) return
       <s:g id="{data($doc/blog/billets/billet[$i]/@num)}">
        <s:rect y="{($i -1)*260+30}" x="40" fill="#e6e6e6" stroke="#000000"
            height="180" width="630"/>
        <s:text font-weight="bold" y="{($i -1)*260+60}" x="60">
          {data($doc/blog/billets/billet[$i]/titre)}</s:text>
        <s:text y="{($i -1)*260+125}" x="55">
          {data($doc/blog/billets/billet[$i]/contenu)}</s:text>
        <s:text font-style="italic" y="{($i -1)*260+195}" x="620" text-anchor="end">
          {concat(concat(data($doc/blog/billets/billet[$i]/auteur),' - '),
           data($doc/blog/billets/billet[$i]/date))}</s:text>
       </s:g> }
    </s:g>
  </s:svg>
```

Figure 6: Transformation du blog en XQuery

Xcerpt est un langage de requêtes pour le web (Berger *et al.*, 2003). La structure de ses programmes se compose en un ensemble de requêtes de construction contenant trois blocs : la construction des données cibles (`CONSTRUCT`), l'origine de données sources (`FROM`) et le *pattern matching* à appliquer sur les données sources (`WHERE`). Tout comme XQuery, Xcerpt permet de structurer les données cibles afin de répondre au problème de la MDW. Cependant ce langage ne permet pas certaines instructions impératives nécessaires lors de transformations de schémas complexes ; c'est pourquoi l'exemple de la figure 4 ne peut être réalisé dans ce langage.

3.3 Discussion

Bien que les principaux langages XSLT et XQuery soient appropriés à la manipulation de données, le principal défaut des approches travaillant au niveau des instances est qu'elles n'assurent pas la conformité des données manipulées par rapport à leur schéma. Pour contourner ce problème, certaines approches permettent de valider les données cibles générées en utilisant un schéma (W3C, 2007b). Le fait de travailler au niveau des instances tout en voulant valider les données utilisées est sujet à erreur puisque cela mélange les instances et leur schéma. Les approches basées sur les schémas ont pour but d'éviter ces problèmes comme l'explique la section 4.

4 Approches basées sur les schémas

Les approches basées sur les schémas considèrent le schéma des données comme l'élément central d'une manipulation de données. Elles permettent ainsi de manipuler les données sources et cibles conformément à leur schéma respectif.

La figure 7 présente le schéma du blog représenté en Relax NG Compact (OASIS,

```
1 start =  element   blog    { attribute nom    {text},
2            element   billets { element billet {Billet}*}}
3 Billet = attribute num     {xsd:integer},
4          element   titre   {text},
5          element   contenu {text},
6          element   date    {text},
7          element   auteur  {text}
```

Figure 7: Le schéma du blog de la figure 4(a)

2002) ; la ligne 1 définit l'élément `blog` comme la racine du schéma, sachant qu'un blog est composé d'un nom (ligne 1) et d'un ensemble de billets (ligne 2). Le schéma SVG est disponible sur le site du W3C[2].

Les approches basées sur les schémas sont basées soit sur la structure des données en tant que types, soit sur la correspondance de schémas, comme le détaillent les deux sections suivantes.

4.1 Les langages de transformation typés

Certains langages de transformation considèrent le schéma comme un ensemble de types utilisables dans un programme, permettant ainsi la conformité des données manipulées. On peut noter que les langages recensés sont des langages fonctionnels. Car ce paradigme de programmation place les fonctions en tant qu'objets de première classe dont les calculs transforment les paramètres d'entrée en données de sortie (Hudak, 1989). De plus, le *pattern matching* permet la sélection des données d'entrées en fonction de critères. Ces avantages ont motivé le développement de nombreux langages fonctionnels manipulant des données structurées ou semi-structurées : XDuce (Hosoya *et al.*, 2003) est un langage fonctionnel au typage statique manipulant des données XML. Ce langage possède un *pattern matching* puissant utilisant les opérations sur les arbres et les expressions régulières.

Développé en parallèle avec XDuce, CDuce[3] est également un langage fonctionnel à typage statique pour XML (Benzaken *et al.*, 2003; Frisch, 2004), partageant de nombreux principes avec XDuce comme son typage fort et son *pattern matching*. CDuce peut être vu comme une extension de XDuce permettant en plus l'utilisation d'itérateurs, la surcharge de fonctions et se voulant plus général (moins orienté XML).

Dans le même registre, le langage OCaml+XDuce, fusion des langages XDuce et OCaml, combine les meilleures propriétés des deux langages (Frisch, 2006).

Circus est un langage typé spécialisé dans la transformation de structures s'adaptant à la manipulation de documents XML (Vion-Dury, 1999; Vion-Dury *et al.*, 2002). Il s'agit d'une extension du lambda-calcul typé non récursif rendant possible la définition de modules. La figure 8 présente un programme Circus de notre exemple dans lequel la définition des types a été volontairement omise.

Le programme Circus (figure 8) se compose d'une première méthode (ligne 2) créant le corps du document SVG. Cette méthode en appelle une seconde (ligne 10) qui construit la représentation SVG des billets du blog.

[2]http://www.w3.org/TR/2002/WD-SVG11-20020108/SVG.xsd

[3]Un prototype est disponible à l'adresse suivante : http://www.cduce.org

```
module blogVersSVG {
const blog: [XMLTree] => [XMLTree]=
 pam x: [XMLTree], y:[XMLTree]
  var s: String.(*( x # [?blog]++ ?x => [|
   blog # <label=%'blog', attr=[{nom=?nomBlog}], sub=?subs>
    => y+= [<label='s:svg', attr=[{xmlns:s='http://www.w3.org/2000/svg'}],
        sub=[<label='s:g', attr=[{font-size='24'}, {fill='#000000'}],
         sub=[<label='s:text', attr=[{x='-280'}, {y='20'}, {text-anchor='middle'},
          {transform='rotate(-90)'}], sub=[nomBlog]>, billets(subs)]>]>]>|]))
const billets: [XMLTree] => [XMLTree]=
 pam x: [XMLTree], y:[XMLTree]
  var s: String.(*( x # [?billet]++ ?x => [|
   billet # <label=%'billet', attr=[{num=?numBil}], sub=[
          <label=%'titre', sub=?titreBil>, <label=%'contenu', sub=?contBil>,
          <label=%'auteur', sub=?autBil>, <label=%'date', sub=?dateBil>]>
    => y+= [<label='s:g', attrs=[{id=numBil}], sub=[<label='s:rect',
            attr=[{y=(pos-1)*260+30},{x='40'},{fill='#e6e6e6'},{width='630'}
            ,{height='180'}, {stroke='#000000'}]>,
            <label='s:text', attr=[{font-weight='bold'}, {y=(pos-1)*260+60},
               {x='60'}], sub=[titreBil]>
            <label='s:text',attr=[{y=(pos-1)*260+125},{x='55'}],sub=[contBil]>,
            <label='s:text',attr=[{font-style='italic'},{y='{(pos-1)*260+195}'},
              {x='620'},{text-anchor='end'}],sub=[autBil+' - '+dateBil]>]>]|]))
```

Figure 8: Transformation du blog en Circus

Du fait de leur puissant *pattern matching* et de leur utilisation d'expressions régulières pour le traitement de données, les langages présentés ont tous la même capacité à réaliser des translations et des transformations de schémas.

4.2 Les approches basées sur la correspondance de schémas

Tout d'abord, définissons la notion de correspondance : une correspondance est un lien entre des éléments de schémas, tandis qu'une correspondance de schémas est un ensemble de correspondances. Les approches basées sur la correspondance de schémas permettent d'établir des correspondances entre des schémas sources et cibles. La spécification de correspondances est un paradigme utilisé originellement dans le domaine des bases de données dont l'enjeu est double (Raffio *et al.*, 2008; Roth *et al.*, 2006) :

1. pouvoir générer, à partir d'une correspondance de schémas, des transformations dans différents langages manipulant des données instances ;
2. capturer la relation entre des schémas pour faciliter la gestion des changements affectant les schémas concernés.

Cette section se focalise sur la capacité des approches présentées à gérer le premier point puisque la problématique concerne la manipulation de données et non l'évolution de schémas.

Tout d'abord, Clio est un outil IBM permettant la définition graphique de correspondances entre un schéma source et un schéma cible (Miller *et al.*, 2000; Miller *et al.*, 2001; Yan *et al.*, 2001; Popa *et al.*, 2002; Haas *et al.*, 2005). Ces correspondances peuvent être enrichies *via* un éditeur d'expressions (expressions arithmétiques par exemple). Clio permet de générer des transformations XQuery, SQL ou XSLT à partir de correspondances de schémas.

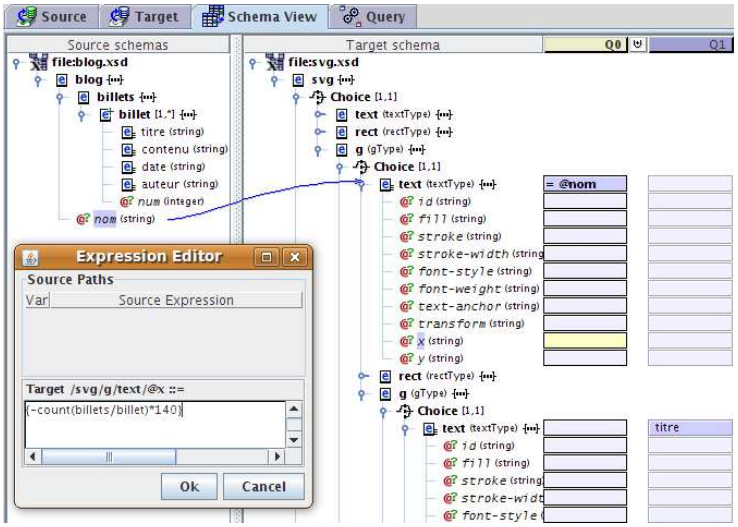


Figure 9: Interface de Clio pour l'exemple du blog

La figure 9 présente l'interface de Clio lors de la définition de la correspondance entre le schéma de la figure 7, contenu dans la partie gauche, et celui de SVG dans la partie droite. Les correspondances sont établies entre les attributs des schémas. Dans notre exemple la première correspondance concerne l'attribut `nom` de l'élément `blog` et un élément SVG `text` ; les autres concernent les éléments `titre`, `contenu`, `date`, `auteur` et `num` des billets, et des éléments SVG `text`. Chaque élément cible peut être défini par une correspondance, ou avoir une valeur fixe en utilisant l'éditeur d'expression (voir la fenêtre à gauche de la figure 9). La transformation d'un blog en un document SVG ne peut pas être réalisée avec Clio puisque certaines opérations impératives, comme l'itération, ne sont pas permises. Il est cependant possible de contourner ce problème en appelant directement les fonctions d'un langage de transformation *via* l'éditeur d'expressions, comme l'illustre la figure 9 avec l'appel de la fonction XSLT `count()`. L'inconvénient de ce processus est la dépendance à un langage de transformation ce qui contredit l'un des principes fondamentaux de la correspondance de schémas.

Influencé par Clio, Clip est un langage graphique réalisant le tracé de correspondances entre des éléments d'un schéma source et d'un autre cible (Raffio *et al.*, 2008). La différence entre ces deux approches est que Clip permet de définir graphiquement la sémantique des correspondances. Le principal inconvénient de Clip est le même que la pluspart des langages graphiques, à savoir le manque de lisibilité et donc de compréhension lorsque le nombre et la complexité des correspondances augmentent.

Malan (*a MApping LANguage*) est un langage de correspondance permettant d'établir des correspondances entre un schéma cible et un schéma source représentés par des diagrammes de classes UML (Blouin *et al.*, 2007; Blouin *et al.*, 2008). L'avantage d'utiliser UML comme format de schéma est qu'il peut représenter les schémas rela-

tionnels de bases de données ainsi que des schémas XML de documents. De plus, UML est un langage de modélisation très largement utilisé. A partir de ces correspondances, il est possible de générer des transformations en XSLT. Malan permet la définition de fonctions afin de spécifier des opérations complexes. Par exemple, si un utilisateur veut mettre en majuscule la première lettre de chaque mot d'un billet (cf. figure 4(a)), il peut définir une fonction `firstLetterUpperCase` qui retourne la chaîne de caractères transformée ; ainsi, il peut appeler cette fonction dans une instruction de correspondance. Il est également possible d'éditer graphiquement des correspondances en utilisant le logiciel *Eclipse* et le plugin *Papyrus*.

```
1 blog.uml -> svg.uml {
2   blog2svg : blog -> svg {
3     24          -> g.font-size        "#000000" -> g.fill
4     -|billets|*140 -> g.text.x      20        -> g.text.y
5     "middle"    -> g.text.text-anchor
6     "rotate(-90)" -> g.text.transform     nom -> g.text.#textContent
7     |billets|   -> |g.g|
8     billets[i]  -> g.g[i], i=[1..|vers|]
9   }
10  billet2g : billet -> g {
11    num -> id                  40        -> rect.x
12    (position(billet)-1)*130+30 -> rect.y
13    "#e6e6e6" -> rect.fill     "#000000" -> rect.stroke
14    180 -> rect.height         630       -> rect.width
15    "bold" -> text[1].font-weight
16    (position(billet)-1)*130+60 -> text[1].y
17    60 -> text[1].x
18    titre -> text[1].#textContent
19    (position(billet)-1)*130+125 -> text[2].y
20    55 -> text[2].x                contenu -> text[2].#textContent
21    "italic" -> text[3].font-style    "end" -> text[3].text-anchor
22    (position(billet)-1)*130+195 -> text[3].y
23    620      -> text[3].x
24    auteur+' - '+date -> text[3].#textContent
25  }
26 }
```

Figure 10: Correspondance de schémas du blog en Malan

La figure 10 présente la correspondance de schéma en Malan pour l'exemple du blog. La ligne 1 définit les diagrammes de classes source et cible utilisés. La correspondance de schémas se compose de deux correspondances: `blog2svg` ligne 2 et `billet2g` ligne 10 ; la première définit le lien entre les éléments `blog` et `SVG`, alors que la seconde spécifie qu'à chaque billet correspond un élément SVG `g` contenant des éléments SVG représentant le contenu de chaque billet.

Malan peut être considéré comme un langage de transformation typé (section 4.1) puisqu'une correspondance de schémas peut être directement instanciée en une transformation par un processeur dédié. Le but principal de Malan est cependant de définir, de manière graphique ou non, la relation entre deux schémas ; des instructions permettent ensuite d'enrichir et de préciser ces correspondances.

4.3 Discussion

Les approches basées sur les schémas répondent certainement le mieux au problème de la MDW. En effet, ces approches garantissent la validité des données et demeurent indépendantes des processus de transformation ; les transformations peuvent ainsi être générées dans différents langages à partir d'une même correspondance de schémas. L'utilisation du principe des correspondances permet également l'application d'opérations importantes sur des sources de données et leur schéma, comme le *schema matching*[4] et le *schema merging*. Selon (Blouin *et al.*, 2008), Malan et Clio seraient complémentaires pour le problème de la MDW ; Malan serait plutôt orienté pour la transformation de schémas tandis que Clio serait mieux adapté à la translation de schémas.

5 Approches IDM

La propriété de base de l'ingénierie dirigée par les modèles (IDM) est de considérer les modèles comme des entités de première classe (Bézivin, 2004). Dans cette approche, chaque modèle est conforme à un métamodèle et représente une partie donnée d'un système, comme décrit dans la figure 11. Un langage décrivant un métamodèle est appelé méta-métamodèle. L'approche IDM a plusieurs buts dont l'indépendance de la plateforme, la séparation, la combinaison et l'identification d'aspects d'un système en développement dans des langages spécifiques, ou encore l'établissement de relations entre ces différents langages dans le but de pouvoir réaliser des transformations entre eux (Bézivin, 2005).

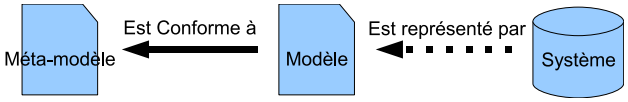


Figure 11: Principe de base de l'IDM

Bien qu'IDM soit appliqué essentiellement à l'ingénierie du logiciel, cette approche peut être appliquée à la MDW. La section suivante donne les différences fondamentales entre les approches basées sur les schémas et celles basées sur les modèles, la section 5.2 présente la transformation de modèles ; la section 5.3 est une discussion sur l'application de l'IDM pour le problème de la MDW.

5.1 Différences entre l'approche basée sur le schéma et l'IDM

L'IDM est nettement plus généraliste que l'approche basée sur les schémas. La représentation de données du web peut se restreindre au format *XML Schema* et au schéma relationnel. Au contraire, l'approche IDM définit des espaces techniques permettant

[4]Le terme *matching* n'est pas à prendre au sens de *correspondance* ; le *schema matching* est une opération visant à trouver de manière (semi-)automatique des correspondances entre des schémas possédant des similitudes.

de représenter des mêmes concepts avec des technologies différentes (XML Schema, DTD, schéma relationnel, etc.). Par exemple, dans l'espace technique XML, décrit dans la figure 12, un document XML est conforme à son XML Schema, qui est lui-même conforme au schéma d'XML Schema. Des langages dédiés (*Domain-Specific language* - DSL) établissent un pont entre les différents espaces, comme les langages TCS et XCS (*Textual Concrete Syntax* et *XML Concrete Syntax*) (Jouault, 2006). Les deux approches possèdent néanmoins des points communs : un schéma correspond à un métamodèle, et un ensemble de données, instance d'un schéma, correspond à un modèle.

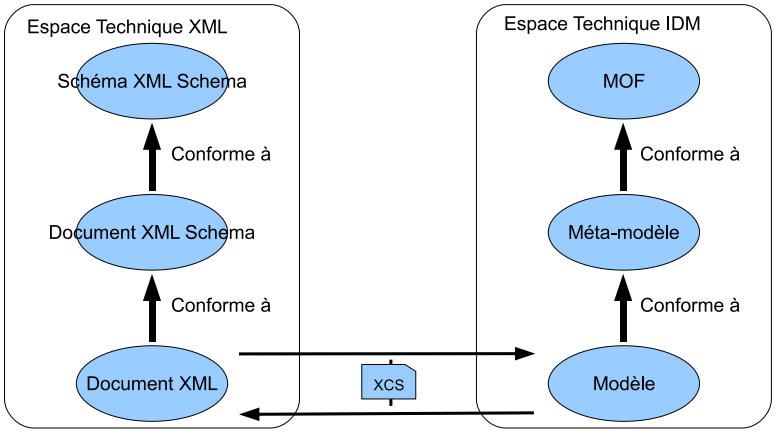


Figure 12: Différences entre l'approche basée sur le schéma et l'approche IDM

Contrairement aux schémas, les espaces techniques assurent une grande flexibilité par rapport à la représentation de données. Cependant, dans le contexte de la MDW, les données sont soient des bases de données, soient des documents XML. Par conséquent, cette flexibilité voit son utilité réduite et provoque alors une contrainte : pour chaque schéma défini, il est nécessaire de spécifier le passage entre le format de ses documents instances et les modèles IDM.

5.2 La transformation de modèles

La transformation de modèles est une opération essentielle dans le cadre de l'IDM visant à répondre aux besoins suivants, selon (Czarnecki *et al.*, 2006) :

- la génération de modèles de plus bas niveau, ou de code, à partir de modèles de plus haut niveau ;
- la création de vues d'un système ;
- la réalisation du *reverse engineering* de modèles de plus haut niveau, à partir de codes ou de modèles de plus bas niveau.

Czarnecki *et al.* (2006) comparent les différents langages de transformation de modèles en se basant sur leurs fonctionnalités. Parmis ces langages, nous pouvons citer ATL et QVT. ATL (*Atlas Transformation Language*) est un langage de transformation pour l'ingénierie des modèles (Jouault, 2006; Jouault *et al.*, 2006b; Bézivin *et al.*, 2003). Il est à la fois déclaratif, afin de permettre la spécification des règles de transformation, et impératif pour la définition d'opérations complexes. Une transformation ATL est un modèle, conforme au métamodèle ATL : elle utilise un métamodèle source `MMs` et un métamodèle cible `MMc` pour créer un modèle cible, conforme à `MMc`, à partir d'un modèle source, conforme à `MMs`. Les trois métamodèles sont eux-mêmes conformes au méta-métamodèle `MOF`.

```
1 module Blog2SVG;
2 create OUT : SVG from IN : Blog;
3 rule blog2svg {
4   from b : Blog!Blog to s : SVG!Svg ( children  <- g1,
5         version <- '1.1', namespace <- 'http://www.w3.org/2000/svg' ),
6     g1 : SVG!G ( fontSize     <- 24, fill  <- '#000000',
7         groupContent <- titreBlog,   groupContent <- billetsBlog ),
8     titreBlog : SVG!Text (
9         position  <- coordTitreBlog,  textAnchor <- 'middle',
10        attribute <- rotateTitreBlog, content     <- b.nom ),
11    coordTitreBlog  : SVG!AbsoluteCoord ( x <- (0-140) * b.billets->size(), y <- 20 ),
12    rotateTitreBlog : SVG!Rotate(angle <- 0-90),
13    billetsBlog     : distinct SVG!G foreach(b1 in b.billets) (
14        id           <- b1.num,
15        groupContent <- rectBillet,    groupContent <- titreBillet,
16        groupContent <- contenuBillet, groupContent <- dateBillet ),
17    rectBillet : distinct SVG!Rect foreach(b1 in b.billets) (
18        position  <- rectPos,       fill    <- '#e6e6e6',
19        stroke    <- '#000000',     size    <- rectDim ),
20    rectPos : distinct SVG!AbsoluteCoord foreach(b1 in b.billets)
21              ( x <- 40, y <- (b.billets->indexOf(b1)-1)*130+30 ),
22    rectDim : SVG!Dimension ( width <- 630, height <- 180 ),
23    titreBillet : distinct SVG!Text foreach(b1 in b.billets)  (
24        position  <- coordTitre, attribute <- fontWeight,   content   <- b1.titre ),
25    coordTitre : distinct SVG!AbsoluteCoord foreach(b1 in b.billets)
26              ( x <- 60, y <- (b.billets->indexOf(b1)-1)*130+60 ),
27    fontWeight : SVG!FontWeight ( bold <- true ),
28    contenuBillet : distinct SVG!Text foreach(b1 in b.billets) (
29        position  <- coordContenu, content   <- b1.contenu ),
30    coordContenu : distinct SVG!AbsoluteCoord foreach(b1 in b.billets)
31              ( x <- 55, y <- (b.billets->indexOf(b1)-1)*130+125 ),
32    dateBillet : distinct SVG!Text foreach(b1 in b.billets) (
33        fontStyle  <- 'italic',   textAnchor <- 'end',
34        position    <- coordDate,  content     <- b1.auteur + ' - ' + b1.date ),
35    coordDate : distinct SVG!AbsoluteCoord foreach(b1 in b.billets)
36              ( x <- 620, y <- (b.billets->indexOf(b1)-1)*130+195 )
37 }
```

Figure 13: Transformation du blog en ATL

La figure 13 présente un programme ATL pour notre exemple du blog[5]. Ce programme est constitué d'une règle de transformation, `blog2svg` définissant les éléments sources et cibles concernés en utilisant les opérateurs `from` et `to`. Cette règle décrit de manière déclarative la structure générale du modèle cible (lignes 5 à 12), ainsi que la transformation des billets en éléments SVG (lignes 13 à 36).

[5]Les détails techniques de cette transformation de modèles (métamodèles, modèles, etc.) sont disponibles à cette adresse : http://gri.eseo.fr/software/malan/blog2svg.zip

Parallèlement à ATL, QVT (*Query View Transformation*) est une recommandation de l'OMG définissant un langage capable d'exprimer des requêtes, des vues et des transformations sur des modèles (OMG, 2005). QVT, comme ATL, est un langage à la fois déclaratif et impératif composé de trois sous-langages appelés *Relations*, *Core* et *Operationnal Mappings*. Jouault *et al.* (2006a) comparent les deux langages en mettant en exergue leurs similitudes afin de permettre l'interopérabilité entre ces deux langages et par conséquent entre leurs outils.

5.3 Discussion

En complément à la transformation de modèles, le tissage de modèle (*model weaving*) permet la spécification de correspondances entre les éléments des différents métamodèles (Fabro *et al.*, 2005; Fabro *et al.*, 2007) ; cela, à la manière de la spécification d'une correspondance de schémas pour les approches basées sur les schémas. L'ensemble de ces correspondances forme un modèle de tissage conforme au métamodèle de tissage. Les buts sont multiples et équivalents à ceux de la correspondance de schémas, à savoir : la génération de transformations de modèles, l'application d'opérations sur des sources de données (*merging* et *matching*), ou encore la capture et la maintenance de la relation sémantique entre des modèles.

Bien que le tissage de modèles permet des opérations essentielles sur les données, en particulier sur les bases de données ; l'approche IDM aspire à répondre à un problème beaucoup plus large que celui des données du web. En conséquence, il est nécessaire de spécifier le lien entre l'espace technique IDM et l'autre utilisé (XML ou relationnel).

6 Conclusion

La manipulation des données est un enjeu majeur pour les composants du web 2.0 tels que les blogs, les flux de syndication ou les RIA (*Rich Internet Application*). Les approches associées ont pour buts de permettre l'interopérabilité entre des sources de données (documents XML et bases de données), l'intégration de données, ainsi que la création de vues.

Dans cet article, nous proposons une classification des différentes approches existantes pouvant être appliquées au problème de la manipulation des données du web. Nous avons notamment divisé les approches en trois groupes : les approches basées sur les instances, celles basées sur les schémas, et celles dirigées par les modèles.

Concernant la translation de données, les approches basées sur la correspondance de schéma semblent les mieux adaptées à ce problème ; l'analyse des correspondances établies entre des schémas homogènes permet à Clio de faciliter le développement d'une translation de données. Le tissage de modèle devrait être également approprié à ce problème puisqu'il se base sur le même concept que la correspondance de schémas.

Une approche cohérente de la transformation de données doit permettre la validité des données manipulées, être indépendante du processus de transformation et avoir un fort pouvoir d'expression. C'est le cas de Malan et des langages de transformation

d'IDM ; cependant IDM tend à répondre à un problème plus général que celui de la manipulation des données du web et apporte donc des contraintes.

La principale caractéristique de l'intégration de données est d'utiliser plusieurs sources de données. Ce problème n'a pas été abordé dans cet article puisqu'il peut être considéré comme de la transformation ou de la translation de données si l'on considère les sources comme un unique ensemble.

Dans un web 2.0 de plus en plus interactif, les approches présentées devrons s'adapter à de nouvelles contraintes telles que la bidirectionnalité des transformations entre une source de données et ses présentations. Malan vise à répondre à ce problème en ayant pour but de générer des transformations actives qui permettent de transformer des documents dans des systèmes interactifs (Beaudoux, 2005).

References

Abiteboul S., " On views and XML", *PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, p. 1-9, 1999.

Abiteboul S., Amann B., Cluet S., Eyal A., Mignet L., Milo T., " Active Views for Electronic Commerce", *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, p. 138-149, 1999.

Abiteboul S., Buneman P., Suciu D., *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers, 2000.

Abiteboul S., Cluet S., Milo T., " Correspondence and translation for heterogeneous data", *Theoretical Computer Science*, vol. 275, n° 1-2, p. 179-213, 2002.

Atzeni P., " Schema and Data Translation", *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, IEEE Computer Society, p. 103, 2006.

Atzeni P., Mecca G., Merialdo P., " Semistructured and structured data in the Web: going back and forth", *SIGMOD Record*, vol. 26, p. 16-23, 4, 1997.

Batini C., Lenzerini M., Navathe S. B., " A comparative analysis of methodologies for database schema integration", *ACM Computing Surveys*, vol. 18, n° 4, p. 323-364, 1986.

Beaudoux O., " XML active transformation (eXAcT): transforming documents within interactive systems", *DocEng '05: Proceedings of the 2005 ACM symposium on Document engineering*, ACM Press, p. 146-148, 2005.

Benzaken V., Castagna G., Frisch A., " CDuce: an XML-centric general-purpose language", *ICFP '03: Proceedings of the eighth ACM SIGPLAN international conference on Functional programming*, ACM Press, p. 51-63, 2003.

Berger S., Bry F., Schaffert S., Wieser C., " Xcerpt and visXcerpt: From Pattern-Based to Visual Querying of XML and Semistructured Data", *VLDB'2003: Proceedings of the 29th international conference on Very large data bases*, 2003.

Bézivin J., " Sur les principes de base de l'ingénierie des modèles", *RSTI-L'Objet*, vol. 10, n° 4, p. 45-157, 2004.

Bézivin J., " On the unification power of models", *Software and Systems Modeling*, vol. 4, n° 2, p. 171-188, 2005.

Bézivin J., Dupé G., Jouault F., Pitette G., Rougui J. E., " First experiments with the ATL model transformation language: Transforming XSLT into XQuery", *OOPSLA Workshop on Generative Techniques in the context of MDA*, 2003.

Blouin A., Beaudoux O., " Mapping Paradigm for Document Transformation", *DocEng '07: Proceedings of the 2007 ACM symposium on Document engineering*, ACM Press, 2007.

Blouin A., Beaudoux O., Loiseau S., " Malan: A Mapping Language for the Data Manipulation", *Proceedings of the 2008 ACM symposium on Document engineering*, 2008.

Coelho J., Florido M., " XCentric: logic programming for XML processing", *WIDM '07: Proceedings of the 9th annual ACM international workshop on Web information and data management*, ACM Press, p. 1-8, 2007.

Czarnecki K., Helsen S., " Feature-based survey of model transformation approaches", *IBM Systems Journal*, vol. 45, n° 3, p. 621-645, 2006.

Drix P., *XSLT fondamental*, Eyrolles, april, 2002.

Fabro M. D. D., Bézivin J., Jouault F., Breton E., Gueltas G., " AMW: A Generic Model Weaver", *Proceedings of the 1ères Journées sur l'Ingénierie Dirigée par les Modèles*, 2005.

Fabro M. D. D., Valduriez P., " Semi-automatic Model Integration using Matching Transformations and Weaving Models", *The 22nd Annual ACM SAC*, 2007.

Fagin R., Kolaitis P. G., Popa L., " Data exchange: getting to the core", *ACM Transactions on Database Systems*, vol. 30, n° 1, p. 174-210, 2003.

Florescu D., Levy A., Mendelzon A., " Database techniques for the World-Wide Web: a survey", *SIGMOD Record*, vol. 27, n° 3, p. 59-74, 1998.

Frisch A., Théorie, conception et réalisation d'un langage de programmation adapté à XML, PhD thesis, Université Paris 7, 2004.

Frisch A., " OCaml + XDuce", *ICFP '06: Proceedings of the eleventh ACM SIGPLAN international conference on Functional programming*, ACM, p. 192-200, 2006.

Gapeyev V., Garillot F., Pierce B. C., " Statically Typed Document Transformation: An Xtatic Experience", *In Workshop on Programming Language Technologies for XML (PLAN-X)*, 2006.

Haas L. M., Hernandez M. A., Ho H., Popa L., Roth M., " Clio grows up: from research prototype to industrial tool", *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, ACM Press, p. 805-810, 2005.

Harren M., Raghavachari M., Shmueli O., Burke M. G., Bordawekar R., Pechtchanski I., Sarkar V., " XJ: facilitating XML processing in Java", *WWW '05: Proceedings of the 14th international conference on World Wide Web*, p. 278-287, 2005.

Hosoya H., Pierce B. C., " XDuce: A statically typed XML processing language", *ACM Transactions on Internet Technology*, vol. 3, n° 2, p. 117-148, 2003.

Hudak P., " Conception, evolution, and application of functional programming languages", *ACM Computing Surveys*, vol. 21, n° 3, p. 359-411, 1989.

Jouault F., Contribution à l'étude des langages de transformation de modèles, PhD thesis, Université de Nantes, 2006.

Jouault F., Kurtev I., " On the architectural alignment of ATL and QVT", *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, ACM Press, 2006a.

Jouault F., Kurtev I., " Transforming Models with ATL", *Satellite Events at the MoDELS 2005 Conference, LNCS 3844*, Springer, p. 128-138, 2006b.

Kay M., *XPath 2.0*, Wiley Publishing Inc., 2004a.

Kay M., *XSLT 2.0*, Wiley Publishing Inc., 2004b.

Kirkegaard C., Møller A., Schwartzbach M. I., " Static Analysis of XML Transformations in Java", *IEEE Transactions on Software Engineering*, vol. 30, n° 3, p. 181-192, 2004.

Kolaitis P. G., " Schema mappings, data exchange, and metadata management", *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ACM Press, p. 61-75, 2005.

Lenzerini M., " Data integration: a theoretical perspective", *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, ACM Press, p. 233-246, 2002.

Miller R. J., Haas L. M., Hernández M. A., " Schema Mapping as Query Discovery", *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc., p. 77-88, 2000.

Miller R. J., Hernandez M. A., Haas L. M., Yan L., Ho C. T. H., Fagin R., Popa L., " The Clio Project: Managing Heterogeneity", *SIGMOD Record*, vol. 30, n° 1, p. 78-83, 2001.

Nguyen T., Srinivasan V., " Accessing relational databases from the World Wide Web", *SIGMOD Record*, vol. 25, n° 2, p. 529-540, 1996.

OASIS, RELAX NG Compact Syntax Specification, Technical report, OASIS, 2002.

OMG, MOF QVT Specification, Technical report, OMG, 2005.

Pietriga E., Vion-Dury J.-Y., Quint V., " VXT: a visual approach to XML transformations", *DocEng '01: Proceedings of the 2001 ACM Symposium on Document engineering*, p. 1-10, 2001.

Popa L., Velegrakis Y., Miller R. J., Hernandez M. A., Fagin R., " Translating Web Data", *VLDB '02: Proceedings of the 28th International Conference on Very Large Data Bases*, p. 598-609, 2002.

Raffio A., Braga D., Ceri S., Papotti P., Hernandez M. A., " Clip: a Visual Language for Explicit Schema Mappings", *Proceeding of IEEE 24th International Conference on Data Engineering*, 2008.

Roth M., Hernandez M. A., Coulthard P., Yan L., Popa L., Ho H. C.-T., Salter C. C., " XML Mapping technology: making connections in an XML-centric world", *IBM Systems Journal*, vol. 45, n° 2, p. 389-409, 2006.

Vion-Dury J.-Y., Circus: un générateur de composants pour le traitement des langages visuels et textuels, PhD thesis, Université Joseph Fourier, Grenoble, 1999.

Vion-Dury J.-Y., Lux V., Pietriga E., " Experimenting with the Circus language for XML modeling and transformation", *DocEng '02: Proceedings of the 2002 ACM symposium on Document engineering*, ACM Press, 2002.

W3C, Scalable Vector Graphics 1.1 Specification, Technical report, W3C, 2003.

W3C, Extensible Markup Language 1.1 Specification, Technical report, W3C, 2006a.

W3C, XQuery 1.0: An XML Query Language, Technical report, W3C, 2006b.

W3C, XML Path Language (XPath) 2.0 Recommendation, Technical report, W3C, 2007a.

W3C, XSL Transformations (XSLT) 2.0 Recommendation, Technical report, W3C, 2007b.

Wallace M., Runciman C., " Haskell and XML: generic combinators or type-based translation?", *ICFP '99: Proceedings of the fourth ACM SIGPLAN international conference on Functional programming*, p. 148-159, 1999.

Yan L. L., Miller R. J., Haas L. M., Fagin R., “ Data-driven understanding and refinement of schema mappings”, *SIGMOD ’01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, ACM Press, p. 485-496, 2001.

Annexe

	Instance	Schéma			IDM
	XSLT/XQuery	XDuce, CDuce, Circus	Clio	Malan	ATL/QVT
Données d'entrées	XML	XML	XML, schéma relationnel	XML	texte
Données de sorties	texte, XML	XML	dépend de la transformation générée		texte
Représentation utilisée	arbre	arbre	relationnelle imbriquée	UML	graphe
Méthode utilisée	langage	langage	outils graphique	langage, outils graphique	langage
Transformation de schémas	oui	oui	limitée	oui	difficilement
Translation de schémas	difficilement	difficilement	oui	difficilement	difficilement
Conformité des données	non	oui	oui	oui	oui
Interaction	il existe des processeurs incrémentaux	non	non	envisage la génération de transformations actives	non

Figure 14: Récapitulatif des approches majeures pour le problème de la MDW