XML_GFD : UN GENERATEUR DYNAMIQUE DE FORMULAIRES XLM VALIDES DTD

MAREDJ Azze-Eddine, KHOUATMI-BOUKHATEM Samia, ADJERAD Halima Douniazed
Centre de Recherche sur l'Information Scientifique et Technique
CERIST
{amaredj, skhouatmi, hadjerad}@mail.cerist.dz

1- Introduction

Actuellement, Internet est l'outil par excellence pour la diffusion et l'échange d'informations. Il est universellement utilisé par les entreprises, les administrations et le grand public et représente aujourd'hui les deux tiers du volume total d'information circulant sur les réseaux. Internet a considérablement évolué depuis les années 90 et va continuer dans les années qui viennent [6][3]. Il est aujourd'hui au centre d'un ensemble de nouveaux besoins : le Web sémantique [2], le e-commerce, le e-gouvernement, la recherche structurée d'informations, etc. Le constat commun à tirer pour accomplir ces évolutions, est que la structuration et la description des données diffusées et échangées sur le Web doivent être rigoureuses et normalisées. HTML a atteint ses limites, de nouvelles technologies sont donc nécessaires. C'est pourquoi qu'un grand espoir est bâti sur la technologie XML qui est pressentie comme une solution aux insuffisances de HTML.

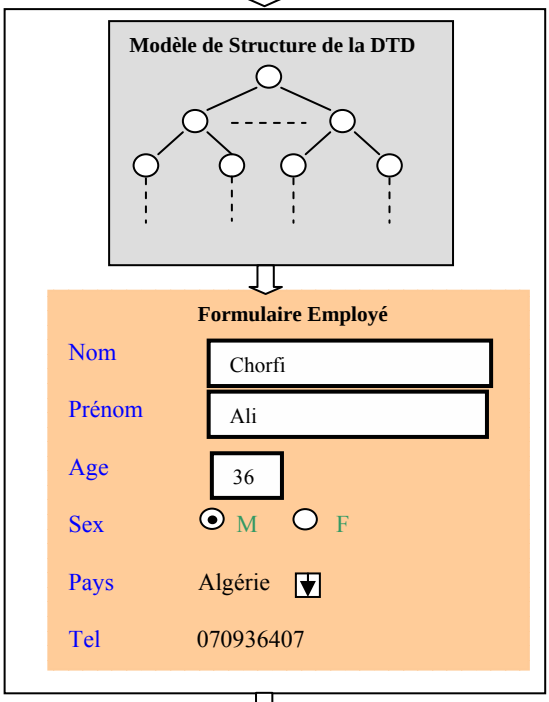
Toutefois, et malgré la profusion d'éditeurs XML qui répondent parfaitement aux exigences d'une production de documents XML, tout le monde s'accorde à dire qu'ils sont dans leur majeure partie complexes et destinés à des usagers ayant des connaissances et des qualifications requises dans le domaine, ce qui peut compromettre l'objectif recherché par le W3C qui stipule qu*'il devrait être facile de créer des documents XML* [1]. En effet, la production de documents XML nécessite des compétences informatiques avérées qu'il faut mobiliser ou former. De plus, le temps et l'attention que nécessite cette opération qui passe par un ensemble de phases (définition d'une structure de document DTD ou schéma, saisie contrôlée, localisation des erreurs éventuelles et leur correction, définition des feuilles de styles, etc.) [4][5][7] peut être

très importants. Il en résulte, pour les entreprises désirant s'approprier cette technologie, un coût de production élevé qui peut leurs représenter une contrainte insurmontable.
Dans cet article nous présentons une contribution au domaine de l'édition de documents XML par la mise en ouvre d'un générateur dynamique de formulaires XML valides DTD, désigné ci-après par XML_GFD.
La suite de l'article est organisée en trois sections. Dans la deuxième, une présentation des principaux concepts retenus pour le développement du système est détaillée. L'architecture fonctionnelle du système est décrite dans la troisième section. La conclusion et les perspectives sont présentées dans la section quatre.

2- XML_GF

XML_GFD est un générateur dynamique de formulaires pour documents XML valide DTD. Partant de la problématique du profil requis, du temps élevé de la production de documents XML et de l'exigence d'une structuration rigoureuse et normalisée des données diffusées et échangées sur le Web, l'idée est d'exploiter la notion de DTD avec le concept des formulaires afin d'offrir un système capable de répondre aux objectifs arrêtés. En effet, nous avons d'une part, la DTD qui définit une structure type, que tout document XML y faisant référence doit respecter. Ceci garantit une production de documents XML normalisés (tous les documents produits ont la même structuration de l'information). D'autre part, il est établi que les formulaires représentent, non seulement, la majeure partie d'interfaces de saisie, mais sont également les plus simples d'utilisation. Un formulaire bien structuré n'exige pas une compétence particulière pour l'agent de saisie, à la limite c'est juste un personnel capable de faire de la saisie. Les contrôles imposés par la DTD, (le respect de l'ordre d'apparition des éléments, de la nature de leur contenu, et de l'oubli d'un attribut obligatoire ou de sa valeur, etc.) sont quasiment réduits à zéro par des contrôles internes au fur et à mesure de la saisie. A la spécification d'une DTD, XML_GFD génère, après analyse de cette dernière, le formulaire de saisie associé. Au cours de la saisie, le système procède aux différents contrôles de validité des informations par rapport aux contraintes de la DTD. Une fois le document XML généré, il est alors valide par construction (Fig.1).

```
< !--  Spécification et Analyse DTD -->
<!ELEMENT employe (personne+)>
<!ELEMENT personne (nom,prenom+,age,sex, pays, tel?)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT age (#PCDATA) >
<!ELEMENT sex (M|F) #IMPLIED>
<!ELEMENT pays (#PCDATA) >
<!ELEMENT tel  (#PCDATA) >
```




```
<?xml version='1.1' encoding='ISO-8859-1' standalone='no'?>
<?xml-stylesheet  href ="C:\style.xsl" type = " text/xsl "?>.
<!DOCTYPE employe SYSTEM "E:\ employe.dtd">
<employe>
        <personne>
                <nom>Chorfi</nom>
                <prenom>Alo</prenom>
                <age>
                        <24-27></24-27>
                </age>
                <diplome>
                        <ingenieur></ingenieur>
                </diplome>
                <tel>26583597</tel>
        </personne>
<employe>
  <personne>
    <nom>Chorfi</nom>
   <prenom>Ali</prenom>
   <age> 36 </age>
   <sex> M </sex>
   <pays> Algérie  </pays>
  <tel> 070936407 <\tel>
  </personne>
</employe>
```

Fig.1 : Génération d'un document XML valide DTD

3- Architecture du système

Etant donné que nous travaillons dans le domaine de l'édition, l'architecture du système doit donc répondre à l'objectif d'une production de documents XML valide-DTD à moindre coût, tout en offrant des fonctions avancées d'un éditeur XML (vérification de la validité de la DTD, réutilisation des formulaires, ajout, suppression, modification d'une élément ou d'un attribut, la personnalisation des formulaires, etc.). Pour ce faire, nous proposons l'architecture fonctionnelle suivante (fig.2) qui s'articule autour de six parties essentielles :

- l'Analyse de la DTD.
- la génération du formulaire.
- l'instanciation d'un document XML.
- la génération du document XML.
- les mises à jour.
- l'adaptation des formulaires de saisie.

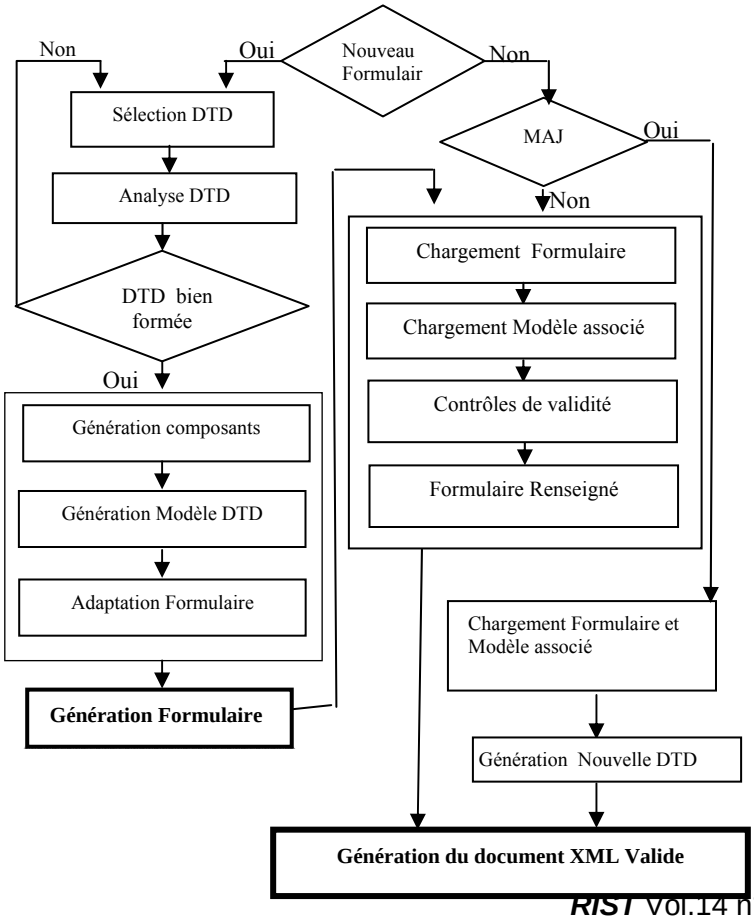


Fig. 2 : Architecture Fonctionnelle

3.1 Analyse de la DTD

Cette partie représente l'amorçage du processus de génération d'un formulaire. C'est à ce niveau que le système vérifie la validité syntaxique de la DTD spécifiée par l'utilisateur. A cet effet, un parseur de DTD a été mis en place (Fig.3). Le processus de génération ne peut se poursuivre qu'une fois la DTD est déclarée valide.

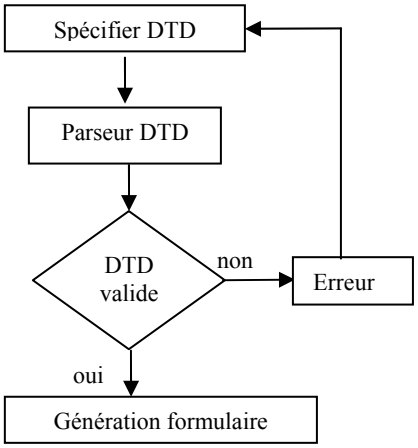


Fig. 3 Analyse d'une DTD

3.2 Modélisation des informations de la DTD

Afin d'uniformiser les traitements (génération et réutilisation des formulaires, génération des instances XML et mises à jours éventuelles), nous avons mis en place un modèle contenant la structure du document et des informations sur les balises de la DTD. Le modèle est une structure d'arbre, ses nœuds contiennent les types de balises de la DTD et un ensemble d'information (tableau1). Le choix de la structure d'arbre est motivé par sa capacité à maintenir les relations hiérarchiques (sémantique) d'un document XML.

Type balise	Informations récupérées
élément	son nom, le nom de son père, son occurrence, son type (#PCDATA, ANY, EMPTY, choix, séquence ou contenu mixte), ses fils et ses attributs.
attribut	son nom, son type (CDATA, ID, IDREFS, NMTOKEN, NMYTOKENS, ENTITY, ENTITIES ou NOTATION), sa déclaration (IMPLIIED ou REQUIRED) et ses valeur (type énuméré) ou valeur fixe.
entité	son nom, son type (simple ou paramétré) et sa valeur de remplacement.

notation	son nom et sa valeur de remplacement

Tableau 1 : informations du modèle de structure

3.3 Génération du formulaire

Comme mentionné, le choix du formulaire comme support de saisie, nous a été dicté, non seulement, par le fait qu'il représente une forme habituelle et conviviale de saisie, mais parce qu'il constitue également un très bon support d'illustration visuelle d'une structure organisée. Afin de répondre au mieux à l'objectif d'une saisie facile et conviviale, il est clair que pour chaque DTD, le système doit générer un formulaire qui reflète au mieux la structure du document cible et l'importance relative des informations à saisir (forme et ordre d'apparition des éléments, leur occurrence, leur encapsulation, couleur, trame, etc.). De là, toute génération standard des formulaires est à écartée. Par conséquent, le système choisit les composants du formulaire (champs de saisie, bouton radio, cases à cocher, etc.) en fonction du type des balises de la DTD (ELEMENT, ATTLIST, ENTITY or NOTATION), de l'information qu'elles contiennent et de leurs liens hiérarchiques (table.2). Pour une mise en évidence de l'information à saisir, l'auteur dispose d'un ensemble d'attributs pouvant caractériser les composants (police de caractère, couleur, trames, dimensions des traits, nombre de lignes, etc.).

Type balise	**Information récupérée**	**Forme correspondante**
ELEMENT	Le nom de l'élément, le nom de son père, son occurrence, son type (simple ou complexe), le nom de ses fils et de ses attributs.	Champ de saisie, bouton radio ou case à cocher.
ATTLIST	Le nom de l'attribut, son type (CDATA, ID, IDREFS, NMTOKEN ou NMYTOKENS, ENTITY, ENTITIES ou NOTATION), sa déclaration (IMPLIIED ou REQUIRED) et ses valeur s'il est de type énumération, par défaut ou valeur fixe.	Champ de saisie ou bouton radio.
ENTITY	Le nom de l'entité, le type de l'entité (simple ou paramétré) et sa valeur de remplacement	Bouton radio ou case à cocher.
NOTATION	Le nom de la notation et sa valeur de	Bouton radio.

	remplacement.	

Tableau 2 : Correspondance balises-composants

3.4 Contrôle de validité d'une instance XML

Pour atteindre l'objectif d'une production de documents XML valides par construction, le système effectue un ensemble de contrôles tout le long du processus d'édition. Pour ce qui est de l'ordre d'apparition des éléments et de leur syntaxe, ils sont conformes à la DTD par construction du formulaire. Le reste est divisé en trois types de contrôles que le système effectue :

- Contrôle lié à la syntaxe XML
  - Les champs de saisie des attributs déclarés *NMTOKEN* et *NMTOKENS* ne doivent comporter que des caractères alphanumériques et quelques autres symboles.
  - Pour un nom XML, le premier caractère doit être alphabétique ou le trait d'union, les caractères de ponctuation ne doivent pas apparaître et il ne doit pas contenir les caractères de balisages « < » « > » « & » « " » « ' », ni des blancs.
- Le contrôle lié aux données

  A ce niveau le système effectue un ensemble de contrôles, qui portent sur l'occurrence des éléments et la déclaration des attributs :
  - L'unicité de la valeur d'un ID.
  - L'appartenance des valeurs introduites de types IDREF et IDREFS à l'ensemble des ID.
  - L'occurrence des éléments.
  - Les types d'attributs déclarés.
  - Les trois premiers caractères ne doivent pas former les chaînes xml, Xml, xMl,…
- Saisie d'une nouvelle instance

  Lors d'une tentative d'une nouvelle saisie d'instance dans le même fichier XML, le système permet l'opération si et seulement si l'élément racine possède un ou plusieurs fils dont au moins un qui possède une occurrence supérieur à un.

3.5 Génération de l'instance XML Valide

Un document XML se compose de texte et d'informations de structure. Les informations de structure (balises) servent à délimiter le texte et donnent une dimension sémantique au contenu. Pour générer un document XML valide DTD, XML_GFD utilise à la fois le fichier formulaire rempli, pour la récupération du texte des éléments et des attributs, et le modèle de structure pour structurer le document XML conformément à la DTD de référence (fig.4).

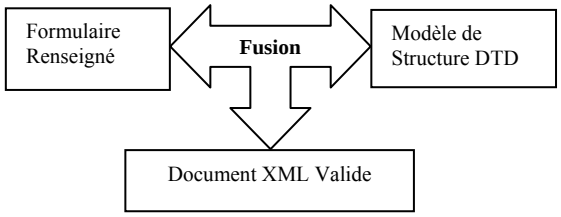


Fig. 4 : XML valide document generation

Un document XML est constitué de la partie prologue et de l'arbre XML, pour générer l'instance XML associée, le système procède comme suit :

3.5.1 ***Génération du prologue***

Il s'agit de la partie introduction d'un document XML. A ce niveau, le système génère automatiquement les trois balises suivantes :

<?xml version=″1.1″encoding=″ISO-8859-1″standalone=″no″?>.

L'attribut *standalone* est automatiquement affecté de la valeur *no* pour signifier que les documents XML générés font référence à une DTD.

<!DOCTYPE *Nom_racine* SYSTEM "*chemin de la DTD*">.

Le système affecte le nom de l'élément racine de la DTD au *Nom_racine* et son emplacement, déjà spécifié, au *chemin de la DTD*.

- Génération de la balise de la feuille de style :

<?xml-stylesheet href ="Chemin_fichier.xsl" type = " text/xsl "?>.

Afin de permettre l'affichage du document XML généré, le système invite l'utilisateur à introduire la feuille de style associée. L'attribut *href* est affecté de la valeur du chemin d'accès à la feuille de style spécifiée et l'attribut *type* reçoit la valeur text/xsl.

3.5.2 Génération de l'arbre XML

Un document XML est une imbrication de balises d'éléments qui doivent apparaître dans l'ordre de leurs déclarations dans la DTD. A ce niveau, le système commence par générer les balises (début et fin) de l'élément racine, elle est obligatoire et unique dans tout le document XML. Le système récupère son nom du fichier modèle de structure et génère ses balises comme suit :

<Nom_racine>
⋮
</Nom_racine>

Suite à quoi, le système entame la génération des balises des éléments. Cette étape se fait selon qu'un élément possède ou pas d'attributs. Dans le cas où il en possède, c'est selon le type et l'occurrence de ses attributs. Pour un élément ne possédant pas d'attribut, la génération des balises et de leurs contenus est effectuée par la récupération de leurs valeurs dans le fichier formulaire renseigné. Le système procède, par la suite, au remplacement des caractères « < », « > », «" », « ‘ », « & » par, respectivement, « &lt ; », « &gt ; », « &quot ;», « &apos ; », « &amp ; ».

Pour l'exemple, d'un élément de type PCDATA ou ANY, la génération se fait de la manière suivante :

Si occurrence est u*nique* : une seule balise portant le nom de cet élément est générée. Elle prend la forme suivante :

<Nom_élément>contenu</Nom_élément>

Si occurrence est *optionnel* : si cet élément possède un contenu, il est comprit entre ses balises, dans le cas contraire rien n'est généré.

<Nom_élément>contenu</Nom_élément>

Si occurrence est *plusieurs* : si l'élément n'a pas de contenu, aucun traitement n'est effectué. Dans le cas contraire, tous les contenus sont présentés entre ses balises comme suit :

<Nom_élément>contenu 1</Nom_élément>
<Nom_élément>contenu 2</Nom_élément>

⋮

<Nom_élément>contenu n</Nom_élément>

Pour les éléments possédant des attributs, la génération de la balise de l'élément s'effectue comme dans le cas précédent. La génération des balise attributs se fait à l'intérieur de la balise de début de l'élément avant la génération du chevron fermant « >» de sa balise. Ce traitement est effectué selon le type de l'attribut :

- CDATA, NMTOKEN, ID et IDREF

La valeur et le nom de l'attribut sont générés comme suit :

Nom_attribut="valeur_attribut"

- NMTOKENS et IDREFS

Les valeurs des attributs sont séparées par des blancs :

Nom_attribut=" valeur _attribut1 valeur _attribut2…… valeur _attribut n"

- Enumération

Les choix effectués sont séparés par des blancs :

Nom_attribut=" valeur_attribut1 valeur_attribu2… valeur_attribut n"

- ENTITY

Pour ce type d'attribut, le système récupère le nom de l'entité sélectionnée. A ce niveau, le système remplace le nom de l'entité par son appel d'entités et le génère comme valeur de cet attribut :

Nom_attribut="&nom_entités;"

- ENTITIES

Le système génère les appels entités qui correspondent aux entités sélectionnées. Elles sont séparées par des blancs.

Nom_attribut="&nom_entités1; "&nom_entités2; …….. "&nom_entités n;"

- NOTATION

La valeur choisie est écrite entre les guillemets de cet attribut :

Nom_attribut="nom_notation"

3.6 Mise à jour

Le système offre un ensemble de fonctions avancées d'édition que nous classons en deux types :

- Mise à jour des contenus

Il s'agit du cas où l'utilisateur souhaite effectuer des mises à jour sur les informations du document déjà saisies. Pour ce faire, l'utilisateur sélectionne juste le formulaire correspondant et le système procède au chargement des données associées sur lesquelles l'utilisateur effectue ses modifications.

- Mise à jour de la structure d'un document XML

Partant de l'objectif d'offrir à l'utilisateur la possibilité de dériver (restreindre, étendre ou changer) un nouveau document à partir d'un document existant. Le système lui offre alors la possibilité de modifier le nom des éléments et des attributs, d'ajouter ou de supprimer un élément ou un attribut. Il est à noter, que pour que le nouveau document XML ainsi généré reste valide, le système génère automatiquement une nouvelle DTD qu'il affecte au nouveau document XML.

3.7 Adaptation des formulaires

Afin d'assurer une ergonomie à même d'aider et de faciliter la saisie des formulaires, le système propose à l'utilisateur la personnalisation de l'interface de saisie avant de la générer. Cette dernière porte sur un ensemble de paramètres à sélectionner se rapportant à la présentation des champs de saisie, du fond du formulaire, à la police et couleur des caractères, etc.

4- Conclusion

Comme mentionné, la production de documents XML nécessite des compétences informatiques avérées qu'il faut mobiliser ou former. De plus, le temps et l'attention que nécessite cette opération qui passe par un ensemble de phases (définition d'une structure de document DTD ou schéma, saisie avec contrôle, localisation des erreurs éventuelles et leur correction, définition des feuilles de styles, etc.) peuvent être très importants. Il en résulte, pour les entreprises désirant s'approprier cette technologie, un coût de production élevé qui peut leur représenter une contrainte financière insurmontable.

La solution que nous proposons dans cet article est un système générant dynamiquement à partie de DTD des formulaires de saisie produisant à la fin des documents XLM valides par construction. Ainsi le coût de la production de documents XML a été largement réduit.

Le système mis en place répond parfaitement à l'objectif du départ et présente des performances très acceptables dans le domaine de l'édition.

Toutefois, et malgré que les DTD sont largement utilisées, il est intéressant d'étendre ce travail au concept des schémas qui offre des mécanismes plus avancés pour le contrôle de données (intégrité, cohérence, occurrence, etc.).

Références :

1. Tim Berners-Lee, *Weaving the Web: the original design and the ultimate destiny of the World Wide Web by its inventor*, Harper, San Francisco, ISBN 0-06-251586-1, 1999.

2. Tim Berners-Lee, James Hendler, Ora Lassila, *The Semantic Web*, Scientific American, May 2001.

3. Tim Berners-Lee, *Web Architecture from 50,000 feet*, http://www.w3.org/DesignIssues/Architecture.html, September 1998.

4. MICHARD, Alain, *XML: Langage et applications.* Eyrolles, 1998.

5. MALER, Eve; EL ANDALOUSSI, Jeanne, *Developing SGML DTDs : From Text to Model to Markup.* Prentice Hall PTR, 1996.

6. Vincent QUINT, *Documents Structurés sur le web* W3C/INRIA, http://www.w3.org/

7. Yves MARCOUX, *Documents structurés XML, SGML, HTML*, GRDS - EBSI Université de Montréal