Conception
Projet Processeur XML

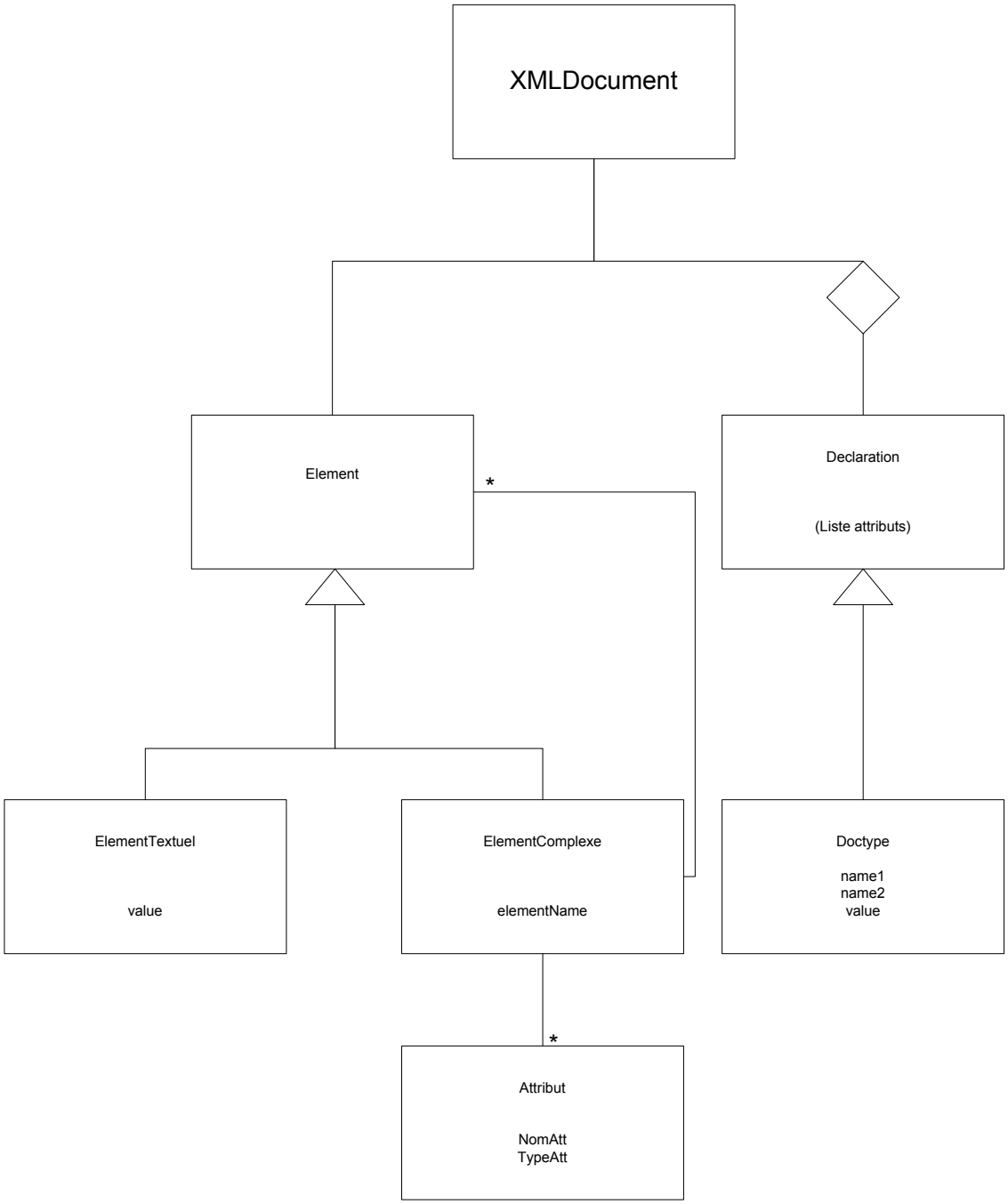
H4212
Etienne Brodu, Chafik Bachetene, Adrien Brochot,
Johann Chazelle, Naby Daouda Diakite, Baptiste Lecornu, Thanh Phan Duc

20 avril 2011

Table des matières

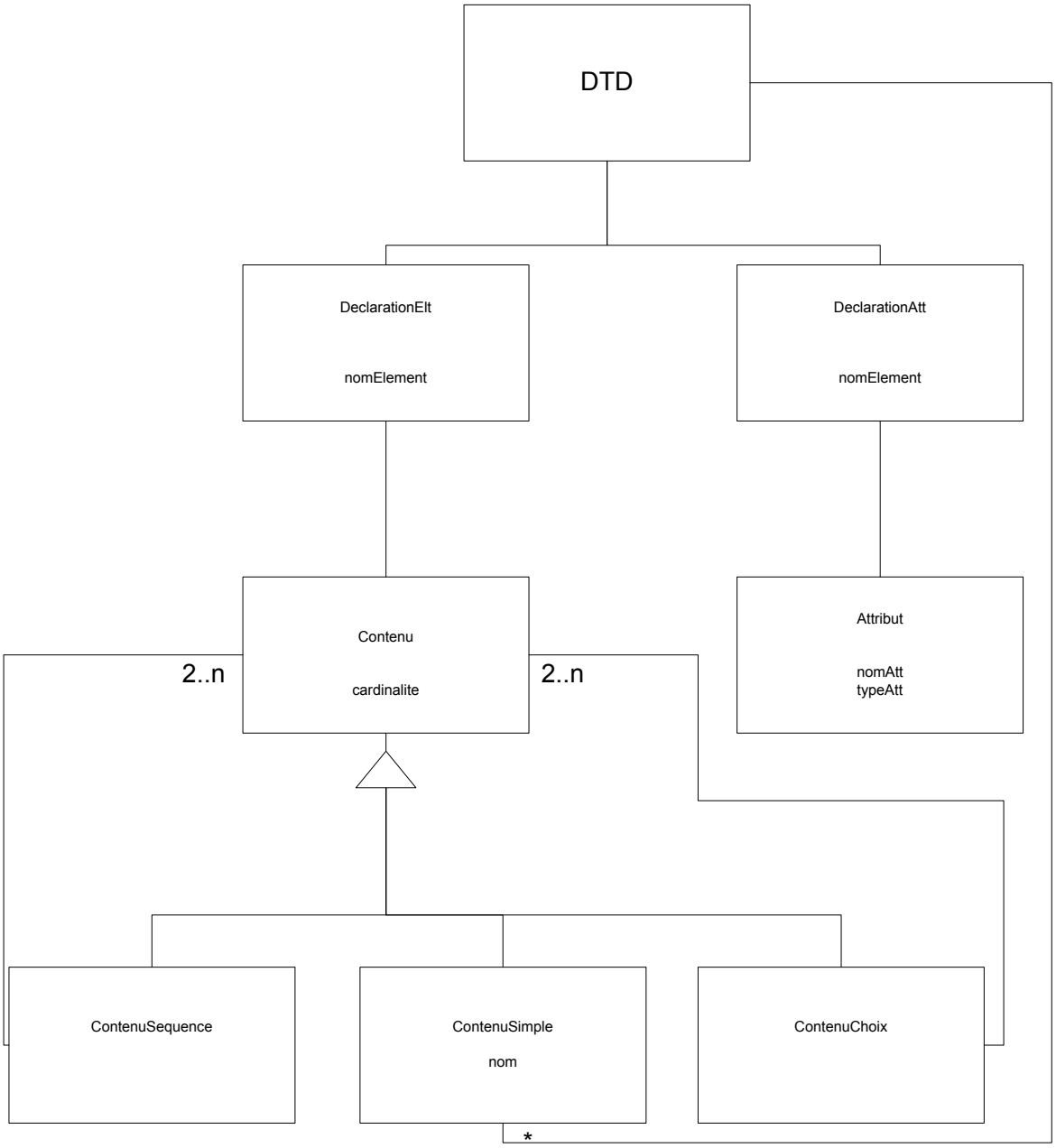
1 Structures de données

1.1 XML



La structure de donnée généré pendant la lecture du XML

1.2 DTD



La structure de donnée généré pendant la lecture de la DTD

2 Algorithmes

2.1 Validation

ALgorithme de validation :

Algorithm 1 validateXML(XML *xml*, DTD *dtd*)

String *exp* ← transformXML(xml) {transformer le fichier XML en une expression a valider}
String *pattern* ← transformDTD(dtd) {transformer le fichier DTD en pattern}
bool *resultat* ← match(exp, pattern) {valider}

Algorithm 2 transformDTD(DTD dtd)

return tranformDeclarationElement(dtd.racine) {on commencer par l'element racine}

Algorithm 3 tranformDeclarationElement(DeclarationElement decEle)

String *result* {transformer recursivement}
if *decEle* est textuel **then**
  result ← "<"+decEle.nom+decATTLIST">decText"</"+decEle.nom+">
else
  result ← "(<"+decEle.nom+decATTLIST">"
  for chaque element fils $filsEle$ **do**
    if pas de contraintes de cardinalité **then**
      result += tranformDeclarationElement(filsEle)
    else
      result += "(" tranformDeclarationElement(filsEle)+cardinalite(filsEle)+")"
    end if
  end for
end if

2.2 Transformation

Algorithme de transformation XSLT :

Explication Il s'agit de parcourir les arbres XML et XSLT de haut en bas de façon indépendante. On commence par pointer sur la racine de l'arbre XML (/). On recherche dans l'arbre XSLT l'élément "template" dont l'attribut "match" est égal à "/". On recopie l'ensemble de ces fils dans l'arbre HTML. Lorsque l'on tombe sur la feuille "apply_templates", on retourne sur l'arbre XML et on vérifie que l'élément actuellement pointé à des fils. Si c'est le cas on pointe sur le premier fils, et on recommence la même opération. Si l'élément pointé n'a pas de fils alors on inscrit dans l'arbre HTML le contenu de l'élément XML. On remonte ensuite dans l'arbre XML et on passe à l'élément suivant jusqu'à la fin du parcours de l'arbre XML.

Exemple

On pointe sur "rapport" dans l'arbre XML

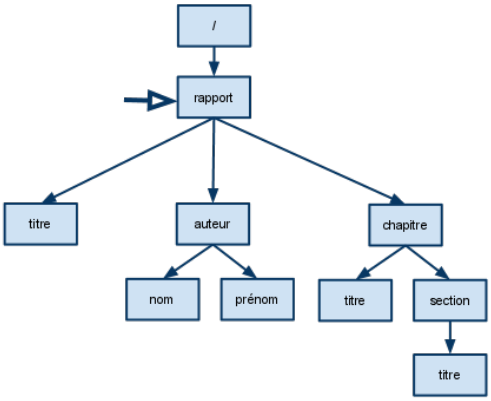
Algorithm 4 transformation(ArbreXML, arbre XSLT)

prendre l'element dont l'attribut est "d̈u XSLT, pour construire la racine de l'arbre de retour.

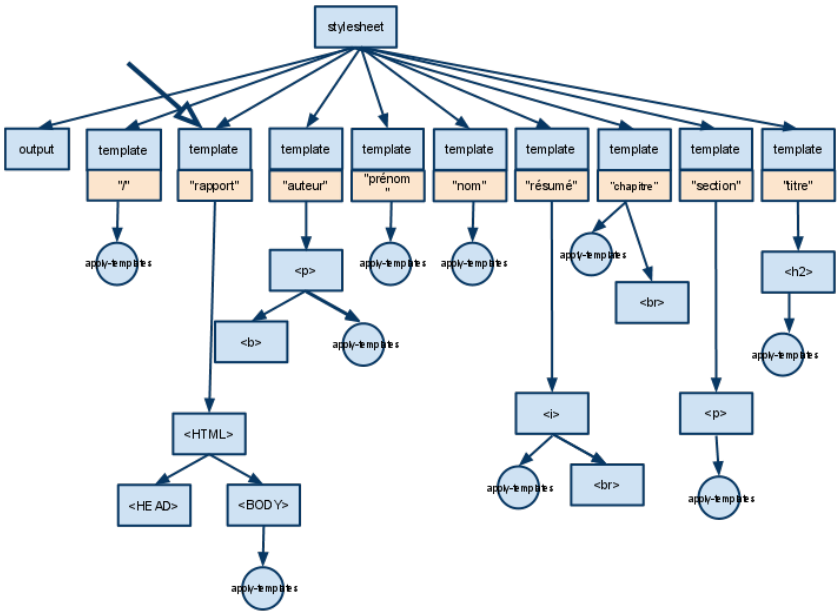
tranformer(root, XSLT("/"))

Algorithm 5 transformer(Element, ArbreXSLT)

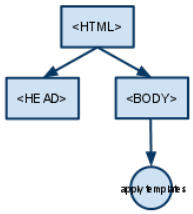
Ajouter à l'arbre le XSL de l'élement.
Chercher l'element ¡xsl :apply-templates/¿, pour le remplacer par soit :
- pour chaque fils, réécrire le contenu XSL de l'élement, et appeler transformation(fils,)
- le contenu XML de l'élément.



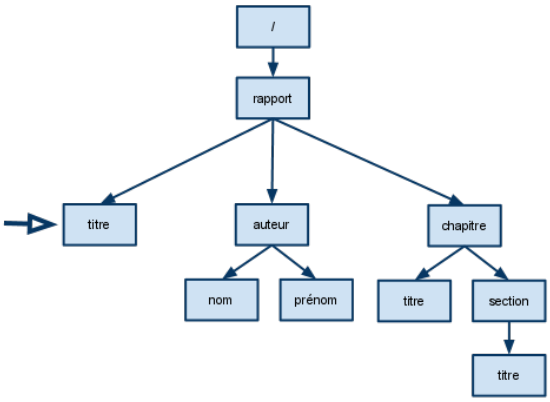
On recherche dans l'arbre XSLT l'élément "template" dont l'attribut "match" vaut "rapport"



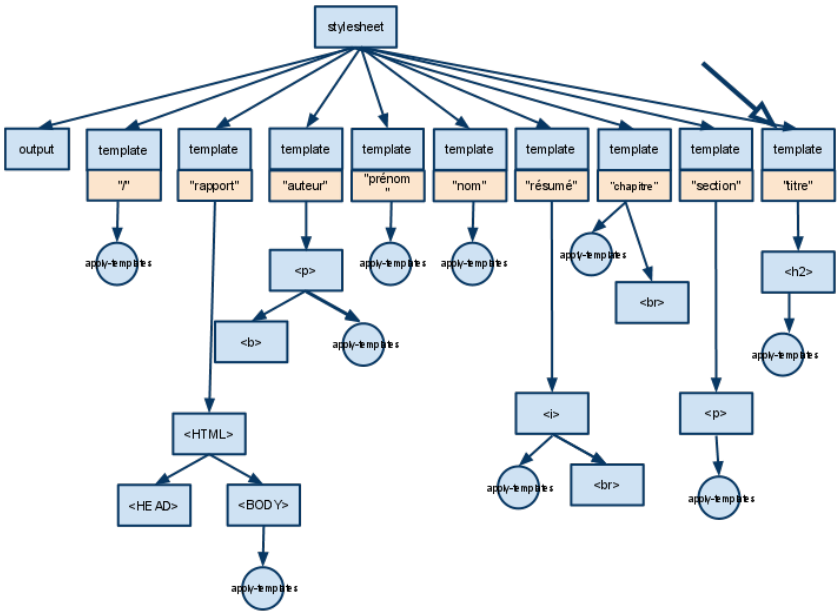
On recopie les éléments fils de cette élément



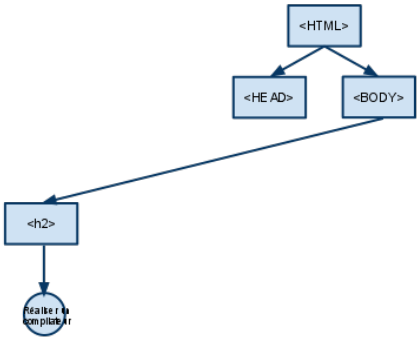
On remarque que cette arbre à une feuille "apply-templates". On retourne donc sur l'arbre XML et on pointe sur le premier fils de "rapport" soit "titre"



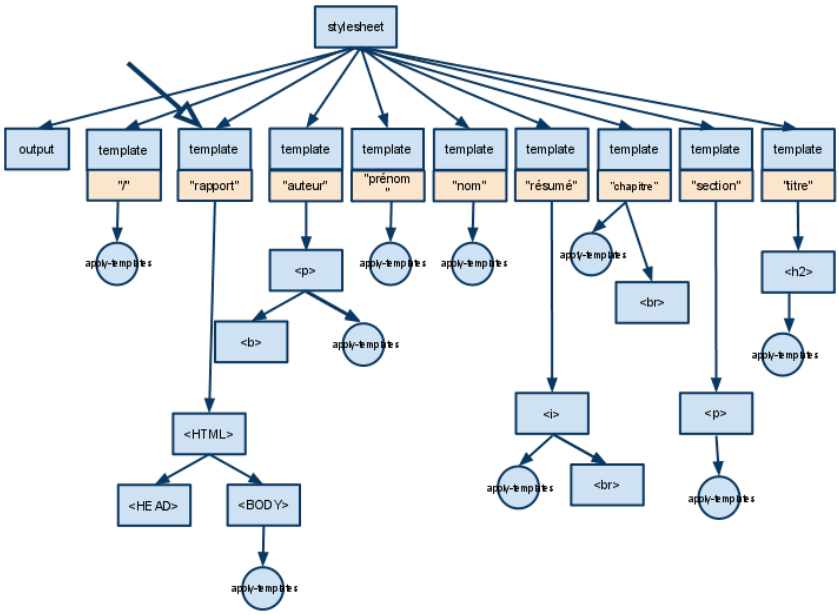
On recherche dans l'arbre XSLT l'élément "template" dont l'attribut "match" vaut "titre"



On recopie les éléments fils de cette élément



Et ainsi de suite jusqu'à l'arbre suivant : On recherche dans l'arbre XSLT l'élément "template" dont l'attribut "match" vaut "rapport"



On recopie les éléments fils de cette élément