Projet 4IF : Développement d'un Processeur XML
Document de Conception

Hexanome 4111 :
Quentin Calvez, Matthieu Coquet,
Jan Keromnes, Alexandre Lefoulon,
Xavier Sauvagnat, Thaddée Tyl

Mars 2012

Destinataire	Version	Etat	Dernière révision	Equipe
Client	7	Validé	30 mars 2012	H4111

Table des matières

1 Objectifs

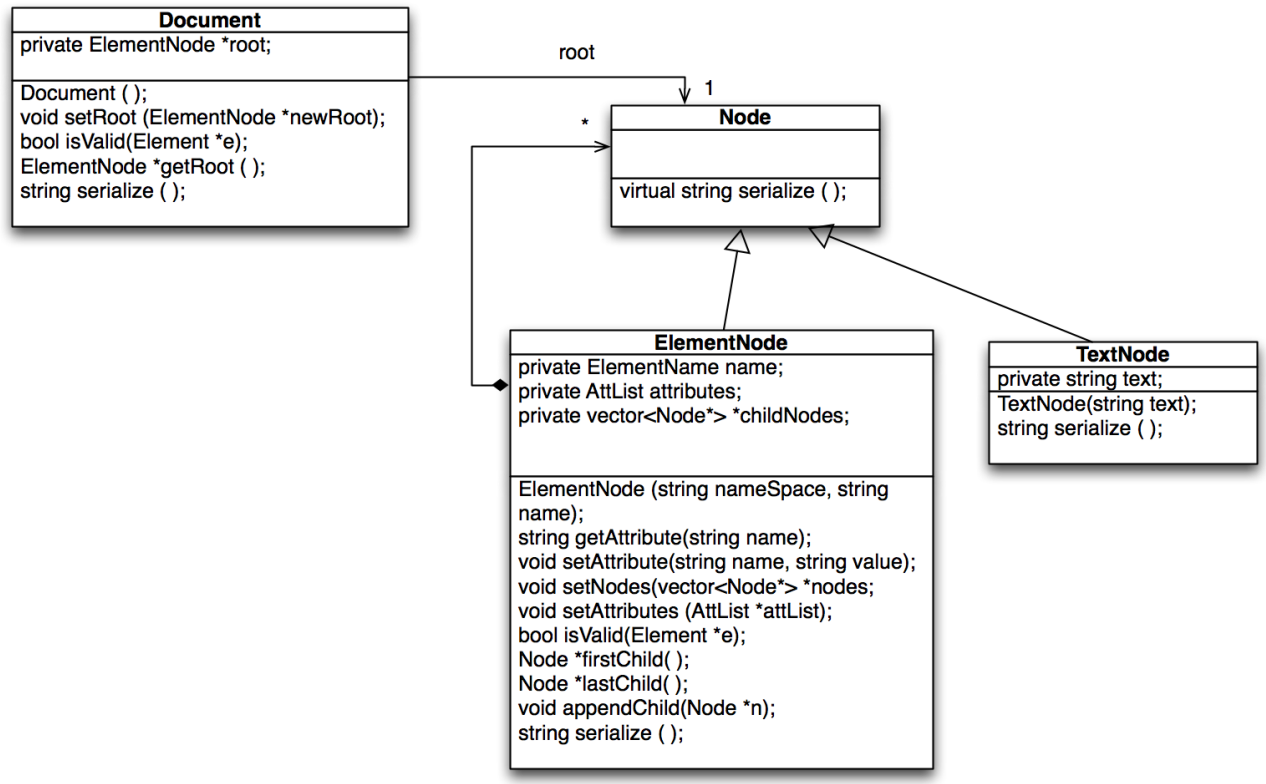
Le but de ce projet était de réaliser un analyseur de documents XML, utilisant les outils flex et bison pour définir une grammaire et construire un parseur. Un document XML est représenté en mémoire selon une structure de donnée définie ci-après, et il peut être validé selon une DTD, dont la structure de donnée est aussi définie ci-après. Un document peut également être transformé grâce à une feuille de style XSLT.

2 Structures de données

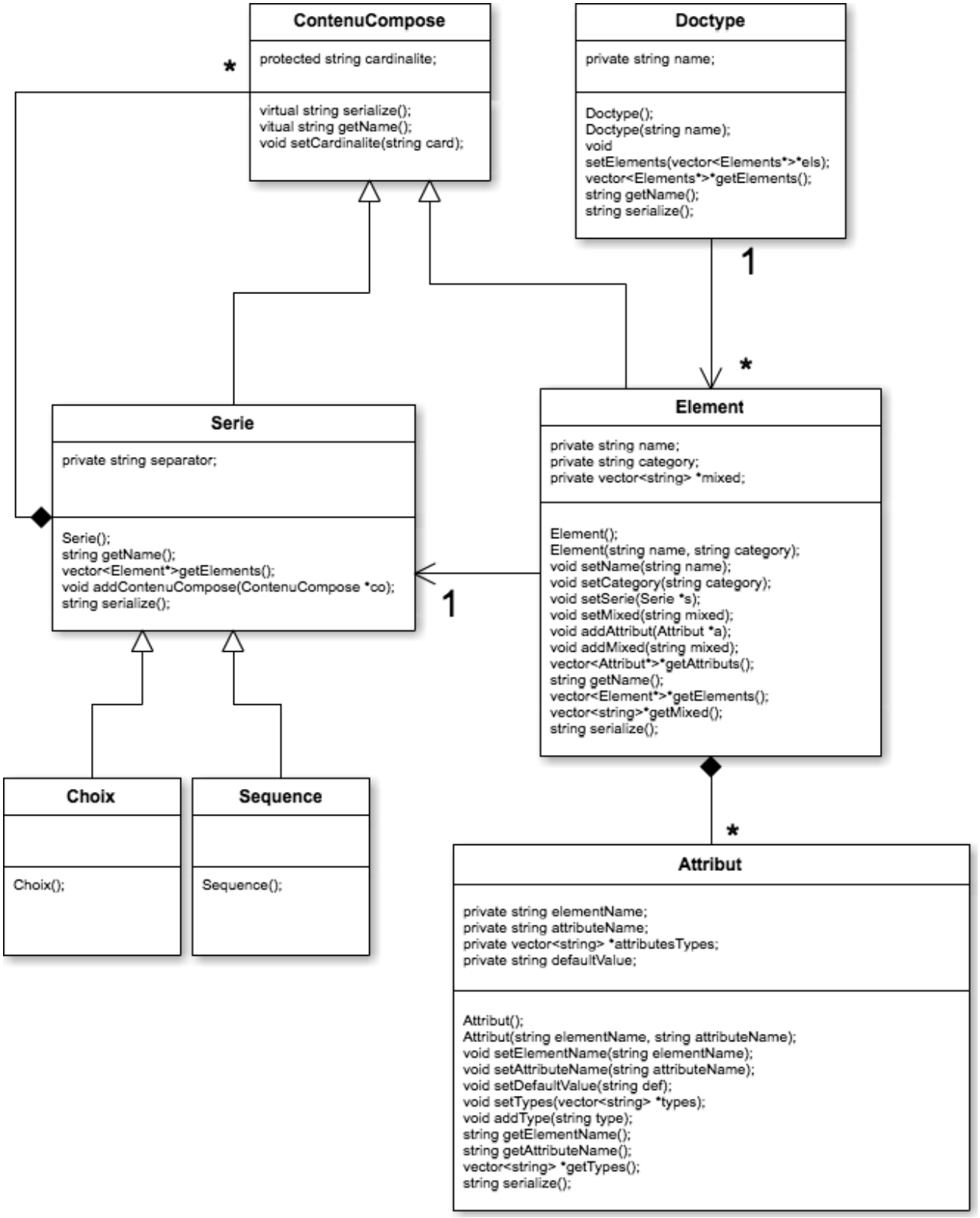
Voici la description des structures de données que nous avons utilisées dans le cadre de ce projet :

2.1 Diagrammes UML détaillés

2.1.1 Modèle XML



2.1.2 Modèle DTD



3 Algorithmes

Voici les algorithmes principaux que nous avons implémenté pour effectuer la validation d'un document XML par une DTD, et pour effectuer sa transformation par rapport à une feuille de style XSLT :

3.1 Algorithmes de Validation

Une validation XML simple a été implémentée de manière algorithmique. Elle fonctionne en parcourant récursivement tous les noeuds d'un document XML, vérifie que chaque noeud est autorisé par la DTD, et que tous ses attributs et sous-éléments aussi :

Algorithm 1 Document : :isValid(DocType)

```
Element ← DocType
return noeudRacine.isValid(Element, DocType)
```

Algorithm 2 ElementNode : :isValid(Element,DocType)

```
if ElementNode.namenotequalsElement.name then
  return false
end if
for all ElementNode.attribut do
  if ElementNode.attributnotinElement.attributs then
    return false
  end if
end for
for all ElementNode.children do
  if ElementNode.childiisElementNodeANDisnotinElement.children then
    return false
  end if
  valid ← ElementNode.child.isValid(Element.child)
  if notvalid then
    return false
  end if
end for
```

3.2 Algorithmes de Transformation

La transformation se base sur une structure XSLT, représentée par le même modèle que pour un document XML classique. L'algorithme parcoure chaque template pour essayer de l'appliquerdepuis le document XML vers le nouveau document.

Algorithm 3 TransformXML(documentXML,documentXSL)

```
resultat ← newDocument
for all template do
  if templatedelaraçine then
    resultat.racine ← transformTemplate(templateRacine, elementNodededocumentXML
  end if
end for
return resultat
```

Algorithm 4 TransformTemplate(nodeXML,template)

```
result ← listedenode
if templaten'apaslenamespacexsl then
  node ← newElementNode(nomdetemplate, attributdetemplate)
  ajoutdenodedansresult
end if
for all enfantdetemplate do
  if enfantestunElementNode then
    if namespacedel'enfantestxsl then
      if nomdel'enfantestapply − templates then
        if untemplates'appliqueàunenfantdenodeXML then
          result ← TransformTemplate(enfantdenodeXML, templateàappliquer)
        end if
      else if nomdel'enfantestvalue − of then
        result ← enfantsdenodeXML
      end if
    else
      result ← TransformTemplate(nodeXML, enfantdutemplate)
    end if
  else
    result ← enfantdutemplate
  end if
end for
return result
```