Modélisation et interopérabilité : Semaine 42, cours 5

Benoît Valiron <benoit.valiron@monoidal.net>

http://inf356.monoidal.net/

Aujourd'hui, validation Relax-NG

Re : Validation avec les DTDs

Appel général :

- Pour les fils
- Caractère des fils (obligatoire...) dans le format du père

```
<!ELEMENT elt format>
```

Appel général :

- Pour l'attribut lui-même
- Caractère explicite

```
<!ATTLIST elt att format caractère>
```

Fils de type texte :

```
<!ELEMENT elt (#PCDATA)>
```

Attribut de type texte :

```
<!ATTRIBUT … … CDATA …>
```

CDATA = “character data” ← ici, < et > n'ont pas de sens spécial...
PCDATA = “parsed character data” ← ici, < et > on un sens spécial

Section CDATA en XML : `<![CDATA[ … ]]>` ← pas de sens spécial pour < et >

Problème avec les DTDs

- Attributs et éléments traités différemment
- Pas de base théorique : fruit de l'histoire
- Peu de souplesse dans les types :
  - ID, IDREF(S), CDATA, …
  - Entier, booléen, flottant, date ?
  - Type complexe : liste, union de type ?
- Pas de gestion des espaces de noms

Méthodes de validation

- XML-Schema (travaux du W3C)
  - Plutôt sur la forme (comme les DTDs)
  - Très complet, mais très complexe...
  - Propose de nombreuses fonctionalités, pas forcément utiles.
- Relax-NG
  - Sur la forme
  - Plus simple, orienté utilisateur
  - Néanmoins base théorique
- Schematron
  - Travail sur le contenu plutôt que sur la forme
- NVDL
  - Combinaison de plusieurs méthode de validation

Dans ce cours : Relax-NG

- Attributs et éléments traités similairement
- Une base théorique !
- Plus de souplesse dans les types :
  - ID, IDREF(S), CDATA, …
  - Entier, booléen, flottant, date
  - Type complexe : liste, union de type
- Gestion des espaces de noms correcte
- Deux syntaxes, une XML et une compacte

Documentation

- http://www.relaxng.org/
- OASIS spec :
  http://www.relaxng.org/spec-20011203.html
- Livre en ligne :
  http://books.xmlschemata.org/relaxng/
  Utilisation du format DocBook !

- Espace de nom :
  http://relaxng.org/ns/structure/1.0

Outils

- Validation : jing
- Traduction : trang (quand c'est possible)
  - Relax-NG ↔ DTD
  - Relax-NG ↔ XML-Schema
  - Relax-NG concis ↔ Relax-NG XML
- http://code.google.com/p/jing-trang/

Format général

```
<?xml version=”1.0”?>
<element name=”personne”>
<!-- commentaire -->
 <attribute name=”né”>
  <text />
 </attribute>
 <attribute name=”mort”>
  <text />
 </attribute>
 <element name=”nom”>
  <element name=”prénom”>
    <text />
  </element>
  <element name=”nomfamille”>
    <text />
  </element>
 </element>
</element>
```

```
element personne {
 # commentaire
 attribute né { text },
 attribute né { text },
 element nom {
  element prénom { text },
  element nomfamille { text }
 }
}
```

```
<!ELEMENT personne (nom)>
<!-- commentaire -->
<!ELEMENT nom (prénom,nomfamille)>
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT nomfamille (#PCDATA)>
<!ATTLIST personne né CDATA #REQUIRED>
<!ATTLIST personne mort CDATA #REQUIRED>
```

Deux formats

- Syntaxe XML

  fichier .rng
- Commentaires :

  <!-- commentaire -->
- Noeud texte :

  <text />
- Noeud élément :

  <element name=”nom”>
- Noeud attribut :

  <attribute name=”nom”>

- Syntaxe compacte

  fichier .rnc
- Commentaires :

  # commentaire
- Noeud texte :

  text
- Noeud élément :

  element nom {...}
- Noeud attribut :

  attribute nom {...}

Notion de père/fils

- Élément `nom` fils de `personne`

```
<element name=”personne”>
 <element name=”nom”>
  ...
 </element>
 ...
</element>
```

```
element personne {
 element nom {
  ...
 }
 ...
}
```

- Attribut `né` fils de `personne`

```
<element name=”personne”>
 <attribute name=”né”>
  ...
 </attribute>
 ...
</element>
```

```
element personne {
 attribute né {
  ...
 }
 ...
}
```

Noeud texte

- Syntaxe XML

<text />

- Exemple :

```
<element name=”prénom”>
 <text />
</element>

<attribute name=”né”>
 <text />
</attribute>
```

- Syntaxe compacte

text

- Exemple :

```
element prénom { text }

attribute né { text }
```

Élément vide

- Syntaxe XML :

```
<element name="a">
  <empty />
</element>
```

- Ou plus court :

```
<element name="a" />
```

- Syntaxe compacte :

```
Element a { empty }
```

Sens : DOIT être VIDE.

`<a />` et `<a></a>` OK

`<a> </a>` pas OK

Élément/Attribut sans rien

```
<element name="a" />
```

→ Sens : sans enfant

```
<!ELEMENT a EMPTY>
```

```
<attribute name="a" />
```

→ Sens : de type texte

```
<!ATTLIST … a CDATA …>
```

(Un attribut n'est jamais vide)

Attention

```
element a {}
```
ERREUR

```
attribute a {}
```
ERREUR

Noeud optionnel

```
<element name=”eltparent”>
 <optional>
  <element name=”eltenfant”>
   <text />
  </element>
 </optional>
</element>
```

```
element eltparent {
  element eltenfant {
    text
  } ?
}
```

```
<element name=”eltparent”>
 <optional>
  <attribute name=”att”>
   <text />
  </attribute>
 </optional>
</element>
```

```
element eltparent {
  attribute att {
   text
  } ?
}
```

Noeud optionnel

```
<element name=”eltparent”>
 <optional>
  <element name=”eltenfant”>
   <text />
  </element>
 </optional>
</element>
```

```
element eltparent {
  element eltenfant {text}?
}
```

```
<!ELEMENT eltparent (eltenfant?)>
<!ELEMENT eltenfant (#PCDATA)>
```

```
<element name=”eltparent”>
 <optional>
  <attribute name=”att”>
   <text />
  </attribute>
 </optional>
</element>
```

```
element eltparent {
  attribute att {text}?
}
```

```
<!ELEMENT eltparent EMPTY>
<!ATTLIST eltparent att CDATA #IMPLIED>
```

Noeud optionnel

```
<element name=”eltparent”>
 <optional>
  <element name=”eltenfant”>
   <text />
  </element>
 </optional>
</element>
```

```
element eltparent {
  element eltenfant {text}?
}
```

```
<!ELEMENT eltparent (eltenfant?)>
<!ELEMENT eltenfant (#PCDATA)>
```

```
<element name=”eltparent”>
 <optional>
  <attribute name=”att” />
 </optional>
</element>
```

```
element eltparent {
  attribute att {text}?
}
```

```
<!ELEMENT eltparent EMPTY>
<!ATTLIST eltparent att CDATA #IMPLIED>
```

Noeud obligatoire en au moins un exemplaire

```
<element name=”eltparent”>
 <oneOrMore>
  <element name=”eltenfant”>
   <text />
  </element>
 </oneOrMore>
</element>
```

```
element eltparent {
  element eltenfant {
    text
  } +
}
```

(Pour les attributs, ça ne fait pas de sens : un attribut ne peut apparaître plus d'une fois)

Noeud obligatoire en zéro ou plus d'exemplaires

```
<element name=”eltparent”>
 <zeroOrMore>
  <element name=”eltenfant”>
   <text />
  </element>
 </zeroOrMore>
</element>
```

```
element eltparent {
  element eltenfant {
    text
  } *
}
```

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Groupe ordonné d'éléments

```
<element name=”eltparent”>
 <group>
  <element name=”eltenfant1”>
   <text />
  </element>
  <element name=”eltenfant2”>
   <text />
  </element>
  <element name=”eltenfant3”>
   <text />
  </element>
 </group>
</element>
```

```
element eltparent {
  element eltenfant1 {text},
  element eltenfant2 {text},
  element eltenfant3 {text}
}
```

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Groupe ordonné d'éléments

```
<element name=”eltparent”>
  <element name=”eltenfant1”>
   <text />
  </element>
  <element name=”eltenfant2”>
   <text />
  </element>
  <element name=”eltenfant3”>
   <text />
  </element>
</element>
```

```
element eltparent {
  element eltenfant1 {text},
  element eltenfant2 {text},
  element eltenfant3 {text}
}
```

Pas de virgule à la fin

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Symmétrie : Exemple

personne1.dtd :

```
<!ELEMENT personne
              (né?,mort?,prénom,nom?)>
<!ELEMENT né (#PCDATA)>
<!ELEMENT mort (#PCDATA)>
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT nom (#PCDATA)>
```

personne1.xml

```
<!DOCTYPE personne SYSTEM “personne1.dtd”>
<personne>
  <né>1912</né>
  <mort>1957</mort>
  <prénom>Alan</prénom>
  <nom>Turing</nom>
</personne>
```

personne2.dtd

```
<!ELEMENT personne EMPTY>

<!ATTLIST personne né CDATA #IMPLIED>
<!ATTLIST personne mort CDATA #IMPLIED>
<!ATTLIST personne prénom CDATA #REQUIRED>
<!ATTLIST personne nom CDATA #IMPLIED>
```

personne2.xml

```
<!DOCTYPE personne SYSTEM “personne2.dtd”>
<personne né=”1912”
          mort=”1957”
          prénom=”Alan”
          nom=”Turing” />
```

Symmétrie : Exemple

personne1.rnc :

```
element personne {
 element né { text } ?,
 element mort { text } ?,
 element prénom { text },
 element nom { text } ?
}
```

personne1.xml

```
<personne>
  <né>1912</né>
  <mort>1957</mort>
  <prénom>Alan</prénom>
  <nom>Turing</nom>
</personne>
```

personne2.rnc :

```
element personne {
 attribute né { text } ?,
 attribute mort { text } ?,
 attribute prénom { text },
 attribute nom { text } ?
}
```

personne2.xml

```
<personne né=”1912”
          mort=”1957”
          prénom=”Alan”
          nom=”Turing” />
```

Symmétrie : Exemple

personne1.rng :

```
<element name=”personne”>
 <optional>
  <element name=”né”><text/></element>
 </optional>
 <optional>
  <element name=”mort”><text/></element>
 </optional>
 <element name=”prénom”><text/></element>
 <optional>
  <element name=”nom”><text/></element>
 </optional>
</element>
```

personne1.xml

```
<personne>
  <né>1912</né>
  <mort>1957</mort>
  <prénom>Alan</prénom>
  <nom>Turing</nom>
</personne>
```

personne2.rng :

```
<element name=”personne”>
 <optional>
  <attribute name=”né”><text/></attribute>
 </optional>
 <optional>
  <attribute name=”mort”><text/></attribute>
 </optional>
 <attribute name=”prénom”><text/></attribute>
 <optional>
  <attribute name=”nom”><text/></attribute>
 </optional>
</element>
```

personne2.xml

```
<personne né=”1912”
          mort=”1957”
          prénom=”Alan”
          nom=”Turing” />
```

Ordre quelconque de fils

```
<element name=”eltparent”>
 <interleave>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </interleave>
</element>
```

```
element eltparent {
  element elt1 {text} &
  element elt2 {text} &
  element elt3 {text}
}
```

Pas d'esperluète à la fin

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Ordre quelconque de fils

```
<element name=”eltparent”>
 <interleave>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </interleave>
</element>
```

```
element eltparent {
  element elt1 {text} &
  element elt2 {text} &
  element elt3 {text}
}


<eltparent>
  <elt1>premier</elt1>
  <elt2>deuxième</elt2>
  <elt3>troisième</elt3>
</eltparent>
```

VALIDE

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Ordre quelconque de fils

```
<element name=”eltparent”>
 <interleave>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </interleave>
</element>
```

```
element eltparent {
  element elt1 {text} &
  element elt2 {text} &
  element elt3 {text}
}


<eltparent>
  <elt1>premier</elt1>
  <elt3>troisième</elt3>
  <elt2>deuxième</elt2>
</eltparent>
```

VALIDE

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Ordre quelconque de fils

```
<element name=”eltparent”>
 <interleave>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </interleave>
</element>
```

```
element eltparent {
  element elt1 {text} &
  element elt2 {text} &
  element elt3 {text}
}


<eltparent>
  <elt1>premier</elt1>
  <elt3>troisième</elt3>
</eltparent>
```

NON VALIDE

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Ordre quelconque de fils

```
<element name="eltparent">
 <interleave>
  <element name="elt1">
   <text />
  </element>
  <optional><element name="elt2">
   <text />
  </element></optional>
  <element name="elt3">
   <text />
  </element>
 </interleave>
</element>
```

```
element eltparent {
  element elt1 {text} &
  element elt2 {text}? &
  element elt3 {text}
}

<eltparent>
  <elt1>premier</elt1>
  <elt3>troisième</elt3>
</eltparent>
```

Ajout du point d'interrogation

VALIDE

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Ordre quelconque de fils

- Pas d'équivalent direct en DTD.
- Peut dans certains cas être simulé :

```
<element name=”eltparent”>
 <interleave>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </interleave>
</element>
```

```
<!ELEMENT eltparent (
      (elt1,elt2,elt3) |
      (elt1,elt3,elt2) |
      (elt2,elt1,elt3) |
      (elt2,elt3,elt1) |
      (elt3,elt1,elt2) |
      (elt3,elt2,elt1))>
<ELEMENT elt1 (#PCDATA)>
<ELEMENT elt2 (#PCDATA)>
<ELEMENT elt3 (#PCDATA)>
```

Un seul élément parmi une liste

```
<element name=”eltparent”>
 <choice>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </choice>
</element>
```

```
element eltparent {
  element elt1 {text} |
  element elt2 {text} |
  element elt3 {text}
}
```

Pas de barre verticale à la fin

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Un seul élément parmi une liste

```
<element name=”eltparent”>
 <choice>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </choice>
</element>
```

```
element eltparent {
  element elt1 {text} |
  element elt2 {text} |
  element elt3 {text}
}


<eltparent>
  <elt1>premier</elt1>
</eltparent>
```

VALIDE

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Un seul élément parmi une liste

```
<element name=”eltparent”>
 <choice>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </choice>
</element>
```

```
element eltparent {
  element elt1 {text} |
  element elt2 {text} |
  element elt3 {text}
}


<eltparent>
  <elt2>deuxième</elt2>
</eltparent>
```

VALIDE

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Un seul élément parmi une liste

```
<element name=”eltparent”>
 <choice>
  <element name=”elt1”>
   <text />
  </element>
  <element name=”elt2”>
   <text />
  </element>
  <element name=”elt3”>
   <text />
  </element>
 </choice>
</element>
```

```
element eltparent {
  element elt1 {text} |
  element elt2 {text} |
  element elt3 {text}
}


<eltparent>
  <elt2>deuxième</elt2>
  <elt3>troisième</elt3>
</eltparent>
```

NON VALIDE

(Pour les attributs, encore une fois, ça ne fait pas de sens)

Combinaison d'indicateurs d'occurence

```
<element name=”personne”>
 <choice>
  <group>
   <element name=”prénom”>
    <text />
   </element>
   <element name=”nom_famille”>
    <text />
   </element>
  </group>
  <element name=”surnom”>
   <text />
  </element>
 </choice>
</element>
```

```
element personne {
  (
    element prénom {text},
    element nom_famille {text}
  ) |
  element surnom {text}
}
```

En syntaxe XML, pas de notation particulière. En syntaxe compacte, utilisation de parenthèses

Combinaison d'indicateurs d'occurence

```
<element name=”personne”>
 <choice>
  <group>
   <element name=”prénom”>
    <text />
   </element>
   <element name=”nom_famille”>
    <text />
   </element>
  </group>
  <element name=”surnom”>
   <text />
  </element>
 </choice>
</element>
```

```
element personne {
  (
    element prénom {text},
    element nom_famille {text}
  ) |
  element surnom {text}
}
```

```
<!ELEMENT personne
   ((prénom, nom_famille) | surnom)>
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT nom_famille (#PCDATA)>
<!ELEMENT surnom (#PCDATA)>
```

Combinaison d'indicateurs d'occurence

```
<element name=”personne”>
 <choice>
  <group>
   <element name=”prénom”>
    <text />
   </element>
   <element name=”nom_famille”>
    <text />
   </element>
  </group>
  <element name=”surnom”>
   <text />
  </element>
 </choice>
</element>
```

```
element personne {
  (
    element prénom {text},
    element nom_famille {text}
  ) |
  element surnom {text}
}
```

```
<personne>
  <prénom>Alan</prénom>
  <nom_famille>Turing</nom_famille>
</personne>
```

VALIDE

```
<personne>
  <surnom>Bobby</surnom
</personne>
```

VALIDE

```
<personne>
  <nom_famille>Turing</nom_famille>
  <prénom>Alan</prénom>
</personne>
```

NON VALIDE

Combinaison d'indicateurs d'occurence

Esperluète

```
<element name=”personne”>
 <choice>
  <interleave>
   <element name=”prénom”>
    <text />
   </element>
   <element name=”nom_famille”>
    <text />
   </element>
  </interleave>
  <element name=”surnom”>
   <text />
  </element>
 </choice>
</element>
```

```
element personne {
  (
    element prénom {text} &
    element nom_famille {text}
  ) |
  element surnom {text}
}
```

```
<personne>
  <prénom>Alan</prénom>
  <nom_famille>Turing</nom_famille>
</personne>
```

VALIDE

```
<personne>
  <surnom>Bobby</surnom
</personne>
```

VALIDE

```
<personne>
  <nom_famille>Turing</nom_famille>
  <prénom>Alan</prénom>
</personne>
```

VALIDE

`interleave` et combinaisons

```
<element name=”eltparent”>
 <interleave>
  <group>
   <element name=”elt1”>
    <text />
   </element>
   <element name=”elt2”>
    <text />
   </element>
  </group>
  <element name=”elt3”>
   <text />
  </element>
 </interleave>
</element>
```

```
element eltparent {
  (
    element elt1 {text},
    element elt2 {text}
  ) &
  element elt3 {text}
}
```

```
<eltparent>
  <elt1>aaa</elt1>
  <elt2>bbb</elt2>
  <elt3>ccc</elt3>
</eltparent> VALIDE
```

```
<eltparent>
  <elt1>aaa</elt1>
  <elt3>ccc</elt3>
  <elt2>bbb</elt2>
</eltparent> VALIDE
```

```
<eltparent>
  <elt3>ccc</elt3>
  <elt1>aaa</elt1>
  <elt2>bbb</elt2>
</eltparent> VALIDE
```

```
<eltparent>
  <elt2>bbb</elt2>
  <elt3>ccc</elt3>
  <elt1>aaa</elt1>
</eltparent> NON VALIDE
```

`text` et combinaisons

- `<text />`
  equivaut à
  `<zeroOrMore><text/><zeroOrMore>`.
- Exemple

```
element para {
  text & element bold { text }
}
```

```
<para>Du texte <bold>en gras</bold> mais pas en
italique</para>
```

- Ceci n'est pas valable (redondance de `text`)

```
element para {
  text & element bold { text } & text
}
```

Élément de contenu mixte

- Première façon :

```
<element name=”paragraph”>
 <interleave>
  <zeroOrMore>
   <element name=”bold”>
    <text />
   </element>
  </zeroOrMore>
  <text />
 </interleave>
</element>
```

```
element paragraph {
  element bold { text } * &
  text
}
```

- Façon plus courte :

```
<element name=”paragraph”>
 <mixed>
  <zeroOrMore>
   <element name=”bold”>
    <text />
   </element>
  </zeroOrMore>
 </mixed>
</element>
```

```
element paragraph {
 mixed {
  element bold { text } *
 }
}
```

Exemple

```
<element name=”section”>
 <group>
  <element name=”title”>
   <text />
  </element>
  <text />
 </group>
</element>
```

```
<section>
  <title>Relax NG</title>
  C'est un bien beau
  format.
</section>
```
VALIDE

```
<section>
  <title>Relax NG</title>
</section>
```
VALIDE

```
<section>
  Le format
  <title>Relax NG</title>
  est bien beau.
</section>
```
NON VALIDE

Exemple

```
element bibliography {
 element desc { text },
 element book {
  attribute isbn { text } &
  element title {
   attribute lang { text }?,
   text
  } &
  element desc { text }? &
  element author {
    (
    element firstname { text },
    element lastname { text }
    ) |
    element name { text }
  }
 }*
}
```

```
<bibliography>
 <desc>Ma bibliothèque</desc>
 <book isbn=”12345”>
  <desc>livre de cours</desc>
  <title>XML</title>
  <author>
   <firstname>Elliotte</firstname>
   <lastname>Harold</lastname>
  </author>
 </book>
 <book isbn=”55543”>
  <title lang=”en”>
    Analysis Now
  </title>
  <author>
   <name>Pedersen</name>
  </author>
 </book>
</bibliography>
```

Exemple : où sont les différences ?

```
element bibliography {
 element desc { text },
 element book {
  attribute isbn { text } &
  element title {
   attribute lang { text }?,
   text
  } &
  element desc { text }? &
  element author {
    (
    element firstname { text },
    element lastname { text }
    ) |
    element name { text }
  }
 }*
}
```

```
<!ELEMENT bibliography (desc,book*)>
<!ELEMENT desc (#PCDATA)>
<!ELEMENT book
      (title,desc?,author)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (
     (firstname,lastname) |
     Name) >
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT name (#PCDATA)>

<!ATTLIST title lang CDATA #IMPLIED>
<!ATTLIST book isbn CDATA #IMPLIED>
```

Exemple : encore plus différent

```
element bibliography {
 element desc {
  element title {text},
  element author {text}
 },
 element book {
  attribute isbn { text } &
  element title {
   attribute lang { text }?,
   text
  } &
  element desc { text }? &
  element author {
    (
    element firstname { text },
    element lastname { text }
    ) |
    element name { text }
  }
 }*
}
```

```
<bibliography>
 <desc>
   <title>Ma bibliothèque</title>
   <author>Moi</author>
 </desc>
 <book isbn=”12345”>
  <desc>livre de cours</desc>
  <title>XML</title>
  <author>
   <firstname>Elliotte</firstname>
   <lastname>Harold</lastname>
  </author>
 </book>
 <book isbn=”55543”>
  <title lang=”en”>
    Analysis Now
  </title>
  <author>
   <name>Pedersen</name>
  </author>
 </book>
</bibliography>
```

Exemple : syntaxe XML

```
<element name=”bibliography”>
 <group>
 <element name=”desc”>
  <element name=”title”><text/></element>
  <element name=”author”><text/></element>
 </element>
 <zeroOrMore>
 <element name=”book”>
  <interleave>
   <attribute name=”isbn” />
   <element name=”title”>
    <optional><attribute name=”lang” /></optional>
    <text/>
   </element>
   <optional><element name=”desc”><text/></element></optional>
   <element name=”author”>
    <choice>
     <group>
      <element name=”firstname”><text/></element>
      <element name=”lastname”><text/></element>
     </group>
     <element name=”name”><text/></element>
    </choice>
   </element>
  </interleave>
 </element>
 </zeroOrMore>
 </group>
</element>
```