

Les Schema XML, une introduction technique

Paris, le -05-2001

Cette présentation a été réalisée par Bruno Chatel et Pierre Attar lors d'un atelier d'une demi-journée dans le cadre de "Forum XML & Intégration e-business", en novembre 2000. Elle a ensuite été complétée pour un atelier d'une journée, toujours dans le cadre des séminaires *Technoforum*, le 4 mai 2001.

Elle se propose de définir les **Schema**, tant d'un point de vue enjeux, avec notamment son positionnement relatif par rapport aux **DTD**, aux **Infoset** et aux **NameSpace**, que d'un point de vue plus technique.

Note : tous les exemples sont *disponibles* [`./demo-schema/`]. Les transparents qui les utilisent sont indiqués par le logo : `[code 3]`. Dans chaque dossier, on trouvera un fichier textuel (`lisez-moi.txt`) indiquant ce qu'il faut regarder dans l'exemple.

Table des matières

- 1. XML et les modèles (p. 2)
- 2. Particularité des fichiers XML utilisés dans les modèles (p. 4)
- 3. Limite des DTD et intérêt des Schema (p. 11)
- 4. Point de vue technique (p. 13)
- 5. Mise en place des Schema (p. 26)

Code 3 : `http://./demo-schema/`

1. - XML et les modèles

Notion de modèle

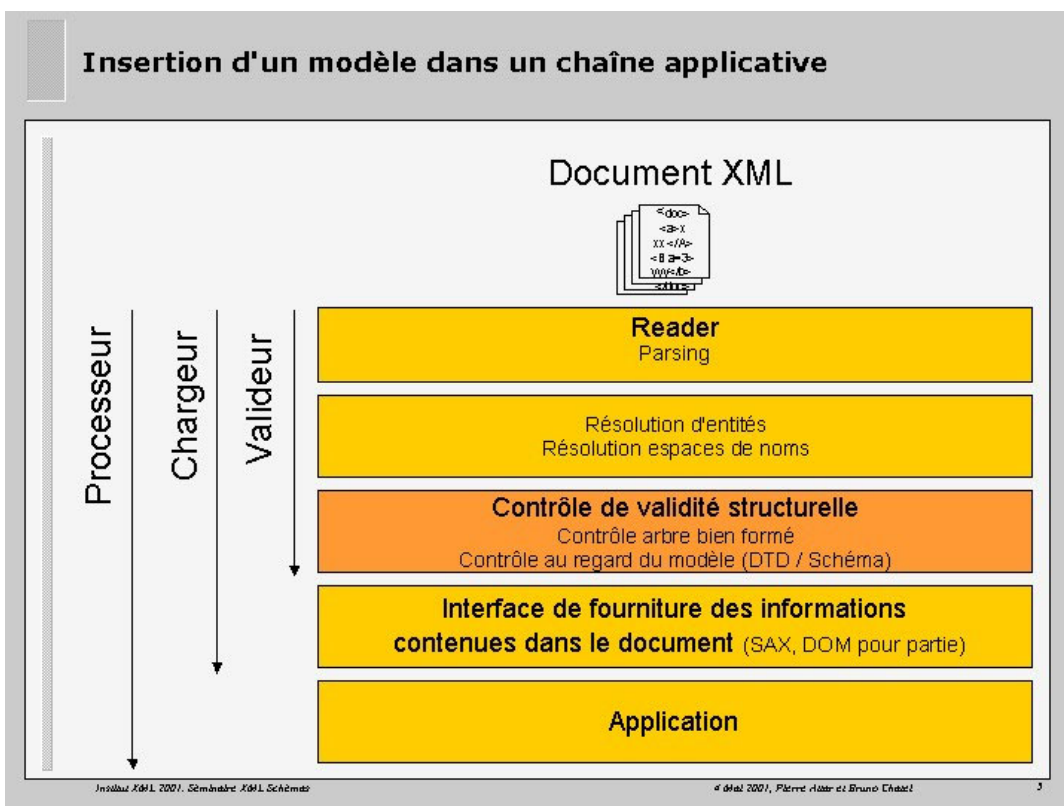
Un double objectif

- **Modéliser un système d'information**
 - 1 - Analyser les informations à traiter
 - 2 - Les modéliser
 - Coopération d'outils de modélisation (UML et document structurés ?)
 - 3 - En déduire un modèle de données
 - Coopération : modèles de documents vs modèles logiques
 - Relève de l'activité traditionnelle d'un projet informatique
- **Décliner des représentations électroniques**
 - XML : une vision hiérarchique de l'information
 - XML : les données respectent un modèle électronique
 - Coopération des modèles XML avec représentations dans des bases de données ou de faits techniques

Modèle de référence vs modèle applicatif

- **Un document structuré est utilisable par des processus**
 - Les processus sont liés à des modèles
 - Les modèles contrôlent les processus

Insertion d'un modèle dans une chaîne applicative



Intérêt d'un modèle applicatif

• Architecture à intégration de composants : 3 niveaux conceptuels

- ...À différencier des architectures marketing 3-tiers d'Internet
 - Présentation, Interaction Client
 - Logique applicative, Middle-tiers, Intégration Composants (x n)
 - Serveur, Gestion des données

XML propose un format standard de données indépendant des choix logiciels...

- basé sur les notions d'objets applicatifs
- auxquels correspondent des composants applicatifs
 - Développement par intégration de composants

Modélisation des données applicatives

- faciliter l'ingénierie applicative (le modèle est un outil à part entière)
- offrir au niveau applicatif un modèle adéquat aux exigences des 3 niveaux

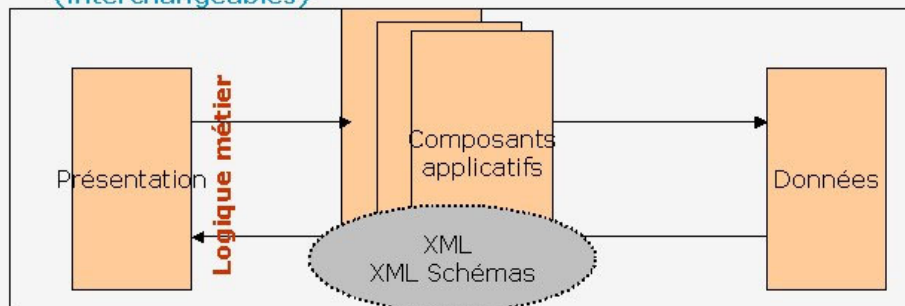
Coopération de modèles

Coopération de modèles

⇒ Modélisation conceptuelle de chacune des couches :

- ◆ **Données : modélisation bases de données**
 - Modèle entité/relation, Merise, MCD, etc.
- ◆ **Logique applicative : modélisation composants**
 - XML Schémas
- ◆ **Présentation**
 - Modèles banalisés (HTML, PDF, XML et XSLT, etc.) ...

⇒ Conception et développement en s'affranchissant des problématiques de modélisation interne aux composants (interchangeables)



Intégration XML 2001. Séminaire XML Schémas

© 2001, Pierre-Henri et Bruno Chérel

7

Modèles électroniques pour XML

Un document XML : un arbre d'objets typés/valués selon un modèle, des objets structurants et des objets données

Le modèle sert à valider le document

- **Modèle syntaxique générique**
 - Arbre bien formé ("*well-formed*")
- **Modèle électronique applicatif**
 - XML : conformité à une DTD, un modèle de document
 - XMLSchema : conformité à un modèle alternatif dont le pouvoir d'expression est augmenté

Un modèle... des modèles

- **Avec les DTDs, le modèle doit être complet et unique**
- **Les Schema introduisent les modèles partiels**
 - sur des fragments de contenus
 - peut-être concurrents, pour des usages différents

2. - Particularité des fichiers XML utilisés dans les modèles

Exemple. Instance XML

Exemple : Instance XML



```
<?xml version="1.0" encoding="UTF-8"?>
<CARNET-ADRESSE DATE-CREATION="2000-10-20" DATE-MAJ="2001-05-02">
  <ADRESSE><IDENTITE><NOM>Dupont</NOM><PRENOM>Gérard</PRENOM></IDENTITE>
  <COORDONNEES>
    <ADRESSE-POSTALE>
      <NUMERO>10</NUMERO><RUE>Boulevard de la Bastille</RUE>
      <CODE-POSTAL>75012</CODE-POSTAL><VILLE>Paris</VILLE>
      <PAYS>France</PAYS>
    </ADRESSE-POSTALE>
  </COORDONNEES>
</ADRESSE>
<ADRESSE><IDENTITE><NOM>Dupont</NOM><PRENOM>Pierre</PRENOM></IDENTITE>
  <COORDONNEES>
    <ADRESSE-POSTALE>
      <NUMERO>14</NUMERO><RUE>rue Saint Saens</RUE>
      <CODE-POSTAL>13001</CODE-POSTAL><VILLE>Marseille</VILLE>
      <PAYS>France</PAYS>
    </ADRESSE-POSTALE>
    <TELEPHONE TYPE="FAX">04 91 00 01 10</TELEPHONE>
    <INTERNET>
      <EMAIL TYPE="PROFESSIONNEL">dupond@mrs.fr</EMAIL>
    </INTERNET>
  </COORDONNEES>
</ADRESSE>
</CARNET-ADRESSE>
```

Infoset

Objectif : différencier la vue sérialisée de la vue sous forme d'arbre

- Une volonté de partager la vision de ce que contient un fichier.
 - Un arbre d'objets typés issu de la désérialisation d'un fichier XML
- Une volonté de partager la vision de ce que contient un fichier
 - Codages alternatifs :
 - `<test type="essai">`, `<test type='essai'>`
 - Jeux de caractères :
 - `essai` vs `essai` vs `essai`
 - Ordonnancement :
 - `<op mode="deb" type="1" >` vs `<op type="1" mode="deb">`
 - Valeurs par défaut
 - `<op mode="deb">` (`<!attlist op mode (1|2) "1">`) vs `<op type="1" mode="deb">`

Principes

- Définir un catalogue d'objets et leurs propriétés
 - *Objet* : `element` ; *Propriétés* : espace de nom et nom, parent, fils et attributs, ...
 - Différenciation entre ce qui est obligatoire et périphérique
- Possibilité d'avoir des outils indépendants de construction d'Infoset

=> Une spécification très controversée pourtant absolument nécessaire.

=> Statut : *working draft* (décembre 2000)

Les objets contenus dans la spécification Infoset

Obligatoires

- document
- éléments
- attributs
- processing instructions
- références à des entités non lues par un parser non validant
- caractères
- notation
- déclaration d'espaces de noms

Périphériques

- DTD
- entités générales déclarées dans la DTD (obligatoire pour les entités non parsées, sinon, périphérique)
- début et fin de localisation d'entité incluse
- début et fin de section définie comme étant `CDATA`
- commentaires

Espaces de noms

Laser in Engineering, 1991, Vol 1, pp 75-88.
Reprints available directly from the publisher.
Photocopying permitted by license only
© 1991 Science Publishers S.A.
Printed in the United Kingdom

Measurements of the main characteristics of a high power laser beam

K.F. BADA,*TH. MANDER+ and P. Gotria*
*I.U.T. Le Paris
+ Treto Design and Research, Technical Centre, Vélizy, France

Abstract : A methodology is described for characterizing a high power laser beam based on the measurement of the diameter of the beam as a function of the distance from the beam waist.

The various sources of error are given as are the precautions to be taken to avoid them. The application of the beam characterization methodology for a pulsed YAG laser beam of a power of 400W is demonstrated and discussed.

KEYWORDS: High-power laser, Beam waist, Maxwell equations.

INTRODUCTION

High power laser of up to several tens of KW or several gigawatts per cm² power density are used to an increasing extent in materials processing (welding, cutting, surface treatment and coating, marking etc.). The temperature of the surface being processed depends, essentially on the characteristics of the material and those of the beam.

Power supply	surface
2W	3cm

2 A RESUME OF THE CHARACTERISTICS OF A LASER BEAM

A laser beam is defined by the maxwell equations and the interference conditions in the resonating cavity. The fundamental equations of its propagation are:

MathML

$$q(z) = q_0 \left[1 + \left(\frac{z\lambda}{\pi q_0^2} \right)^2 \right]^{-1/2}$$

⇒ Faire coopérer les noms de différentes structures XML

Intésumé XHTML 2001. Sommaire XHTML Schémas
© 2001, Pierre-Alain et Bruno Cheval
13

Utilisation des NameSpace

Utilisation des espaces de noms

⇒ **Déclaration**

```

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
>
```

⇒ **Utilisation dans le corps document**

```

<xsl:template match="para">
  <xsl:choose>
    <xsl:when test="$generation='papier'">
      <fo:block><xsl:apply-templates/></fo:block>
    <xsl:when>
    <xsl:otherwise>
      <xhtml:p><xsl:apply-templates/></xhtml:p>
    <xsl:otherwise>
    </xsl:choose>
  </xsl:template>
```

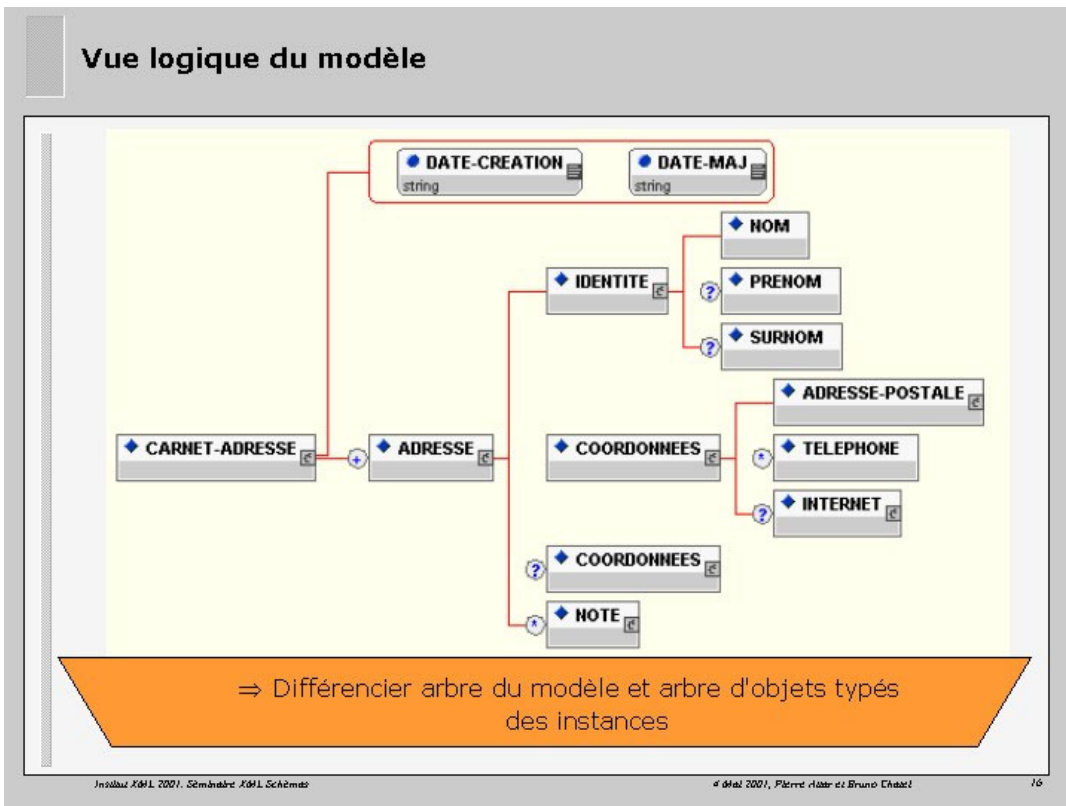
⇒ Le modèle sous-jacent n'est pas obligatoirement explicite
 ⇒ Non utilisable dans les DTD

Intésumé XHTML 2001. Sommaire XHTML Schémas
© 2001, Pierre-Alain et Bruno Cheval
14

Code 4 : <http://demo-schema/d1-s10/>


2.1. - Les DTD

Vue logique du modèle



DTD correspondante [code 5]

DTD correspondante



```

<!ELEMENT CARNET-ADRESSE (ADRESSE+) >
<!ATTLIST CARNET-ADRESSE
    DATE-CREATION CDATA #REQUIRED
    DATE-MAJ CDATA #REQUIRED >
<!ELEMENT ADRESSE
    (IDENTITE,COORDONNEES,COORDONNEES?,NOTE*)>
<!ELEMENT IDENTITE (NOM,PRENOM?,SURNOM?)>
<!ELEMENT NOM (#PCDATA)>
<!ELEMENT COORDONNEES
    (ADRESSE-POSTALE,TELEPHONE*,INTERNET?)>
<!ELEMENT TELEPHONE (#PCDATA)>
<!ATTLIST TELEPHONE
    TYPE (STANDARD|DIRECT|FAX|MOBILE) #REQUIRED>

```

Modèles de contenus

Valuation : attributs

Organisation de contenus

Occurrences
Optionnalités

Typage de données

Intésumé XML 2001. Séminaire XML Schémas
© 2001, Pierre-Alain et Bruno Chérel
17

Le langage XML : l'élément

Le langage XML : l'élément

Déclaration d'un élément par son contenu


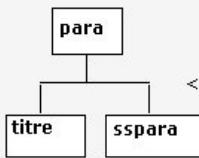
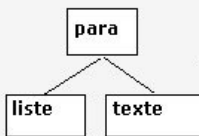
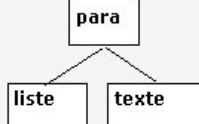
contenu déclaré	modèle de contenu ()
. vide <pre style="font-family: monospace; font-size: 0.9em; margin: 0;"> <!ELEMENT imag EMPTY > </pre>	. textuel <pre style="font-family: monospace; font-size: 0.9em; margin: 0;"> <!ELEMENT titre (#PCDATA) > occurrence implicite . groupe <!ELEMENT val (déclarée mesurée) > . groupe et/ou textuel <!ELEMENT p (#PCDTATA val)*> </pre>

Intésumé XML 2001. Séminaire XML Schémas
© 2001, Pierre-Alain et Bruno Chérel
18

Code 5 : <http://demo-schema/d2-s17/>

Le langage XML : ordonnancement de contenus





Le langage XML : ordonnancement de contenus

contenu ()		<code><!ELEMENT para (titre, sspara+) ></code>
séquence ,		<code><!ELEMENT para (titre , sspara+) ></code>
choix		<code><!ELEMENT para (liste texte) ></code>
any		<code><!ELEMENT para ANY ></code>

Intésumé XML 2001. Séminaire XML Schémas © 2001, Pierre-Alain et Bruno Chérel 19

Le langage XML : indication d'occurrences

Le langage XML : indication d'occurrences

obligation 1 obligatoire		<code><!ELEMENT chap (titre, sspara+) ></code>
option ? 1 optionnel		<code><!ELEMENT sspara (titre?, texte+) ></code>
répétition + 1 ou plusieurs		<code><!ELEMENT table (lem, déf)+ ></code>
option et répétition * 0, 1 ou plusieurs		<code><!ELEMENT para (titre, sspara+) ></code>

Intésumé XML 2001. Séminaire XML Schémas © 2001, Pierre-Alain et Bruno Chérel 20

Attributs : une méthode de qualification

Attribut : une méthode de qualification

Déclaration de liste d'attributs

```
<!ELEMENT image EMPTY >
<!ATTLIST image source (diapo, photo)    ``photo``
        idf ID                            #REQUIRED
        lien IDREF                         #IMPLIED>
```

nom d'élément
nom d'attribut
valeur par défaut
type

valeur
#FIXED
#REQUIRED
#IMPLIED

énumération de valeurs
CDATA
ENTITY (IES)
ID
IDREF (S)
NMTOKEN (S)
NOTATION

Jérôme Lhôte, 2007. Séminaire XML Schémas © 2007, Pierre-Henri et Bruno Chérel

Entité et DTD

Deux outils : ingénierie des DTD (paramètre) ou des documents (générale)

Fonctionnement de l'entité paramètre

• Déclaration

- Interne :
 - `<!ENTITY % tousParagraphes "P|Table|Liste|InterTitre">`
- Publique :
 - `<!ENTITY % table PUBLIC "-//OASIS//DTD Exchange Table Model 19960430//EN" "calstbl.ent">`

• Utilisation

- `<!ELEMENT CHAP (Titre, ((%tousParagraphes;)+) >`
- `%table;`

Intérêt de l'entité paramètre

- **Support à l'ingénierie des modèles**
 - Réutilisation par factorisation
 - Obligation de cohérence
- **Limites levées par les espaces de noms**
 - Risques de collision

L'entité générale propose le même principe pour les *documents*

- **Fragmentation et travail collaboratif**
- **Définies dans les DTD : définitions partagées vs surcharge locale**
 - `external subset` vs `internal subset`

3. - Limite des DTD et intérêt des Schema

Limite des DTD

DTD (Document Type Definition)

- **Double rôle :**
 - Modèle selon une organisation hiérarchique
 - Définition des éléments, attributs, contenus
 - Définition d'entités
 - Mécanisme d'inclusion (interne, externe, paramètre) particulièrement utile pour les opérations de modularisation et de réutilisation
- **Héritage du monde SGML et simplification XML 1.0**

Limites des DTD

- **Syntaxe spécifique**
- **Typage**
 - Pas de possibilité de typer les contenus
 - Typage faible des valeurs d'attributs (orienté *document* plutôt que *XML-Data*)
- **Pas de modélisation partielle : la modélisation doit être complète**
- **Pas d'intégration des espaces de noms**

Avenir des DTD

- **Remplacement : la non-prise en compte d'espace de noms condamne les DTD**
- **Coopération : les Schémas ne permettent pas la définition d'entités externes**

Schema correspondant [code 6]

Schéma correspondant



```
<xs:schema xmlns:xs="http://www.w3.org/2000/10/XMLSchema">
  <xs:element name="CARNET-ADRESSE">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="ADRESSE" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="DATE-CREATION" use="required" type="xs:date"/>
      <xs:attribute name="DATE-MAJ" use="required" type="xs:date"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="ADRESSE">
    <xs:complexType >
      <xs:sequence>
        <xs:element ref="IDENTITE"/>
        <xs:element ref="COORDONNEES" minOccurs="1" maxOccurs="2"/>
        <xs:element ref="NOTE" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="IDENTITE">
    <xs:complexType >
      <xs:sequence>
        <xs:element ref="NOM"/>
        <xs:element ref="PRENOM" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="SURNOM" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  ... </schema>
```

Intégré XSL 2001. Séminaire XSL Schémas

© mai 2001, Pierre-Henri et Bruno Chérel

23

Objectifs des Schema

Définir un nouveau mécanisme de support à la modélisation

- **Reprenant les acquis des DTD en termes de définition de modèles**
 - Arbres d'objets typés et valués
- **Permettant d'exprimer des contraintes fortes**
 - Typage de données plus puissant et évolutif
- **Utilisant XML pour définir les modèles eux-mêmes**
 - Factorisation des outils et des méthodes

En plus

- **Permettre de définir des contraintes incomplètes**
 - Complémentarité aux DTD : outil de validation de données
- **S'intégrer à l'ensemble des spécifications W3C**
 - Prise en compte des espaces de noms
- **Se rendre indépendant du format XML sérialisé**
 - La validité s'applique à une structure logique basée sur les Infoset

=> Travail en cours au sein du W3C

=> *Proposed Recommendation*, mars 2001

Code 6 : <http://demo-schema/d3-s25/>

4. - Point de vue technique

Un Schema est un document XML

Doit être conforme à **XML 1.0**

Sa syntaxe est liée à un espace de noms

- doit être "Well-Formed"
- doit être valide au regard de sa spécification
 - en utilisant le Schema des Schema
 - en utilisant la DTD de Schema

Relation Schema/document

- Un Schema peut être identifié dans une instance
 - `schemaLocation`
- L'association d'un Schema à une instance peut être réalisée par programme

Profiter de toute l'ingénierie XML pour faire de la qualité sur les Schema

Fonctionnalités

Fonctionnalités

- ⇒ Un élément a un contenu... défini selon un modèle de contenu
 - † Simple
 - **Contenus typés** : caractères, dates, etc.
 - † Complexe
 - **Organisation d'éléments entre eux**
Séquentielle, groupée... avec notions d'occurrences (`minOccurs`, `maxOccurs`)
 - **Contenus mixtes**, avec des inter-relations entre éléments et caractères de texte
- ⇒ L'emboîtement des modèles de contenus définit une structure hiérarchique
- ⇒ Un élément peut avoir des attributs
- ⇒ Le contenu peut être défini de façon indépendante de l'élément
 - † Définitions de types (Simples ou Complexes) réutilisables

```
<xs:element name="NOM" type="xs:string" />
<xs:element name="IDENTITE">
  <xs:complexType mixed="false">
    <xs:sequence>
      <xs:element ref="NOM" />
      <xs:element ref="PRENOM"
        minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="PARA" type="bloc" />
```

Intésumé XML 2001, Séminaire XML Schémas

© mai 2001, Pierre-Alain et Bruno Chabot

29

4.1. - Modèles de contenus : contrôle de l'arbre d'objets typés et valués

Global versus local

Global versus local

⇒ Utilisation de déclaration locales

```
<xs:element name="CARNET-ADRESSE">
  ... <xs:element name="ADRESSE"
    minOccurs="1" maxOccurs="unbounded">
    ... <xs:element name="IDENTITE">
      ... <xs:element name="NOM" type="xs:string"/>
      <xs:element name="PRENOM"
        type="xs:string" minOccurs="0" />
    </xs:element>
  </xs:element>
  <xs:attribute name="DATE-MAJ" use="required" type="xs:date"/>
</xs:element>
```

⇒ Utilisation de déclaration globales

```
<xs:element name="ADRESSE">
  ... <xs:element ref="IDENTITE"/> ...
</xs:element>
<xs:element name="IDENTITE">
  ...
  <xs:element ref="NOM"/>
  <xs:element ref="PRENOM" minOccurs="0" maxOccurs="1"/>
  ...
</xs:element>
<xs:element name="NOM" type="xs:string"/>
```

S'applique aux déclaration d'élément, d'attributs et de types

Ordonnancement

Ordonnancement

⇒ Sequence : séquence ordonnée d'éléments (ET logique)

```
<element name="adresse"><complexType>
  <sequence>
    <element name="IDENT" type="identType" />
    <element ref="ADRESSE-POSTALE" /><element ref="ADRESSE-TEL" />
    <element ref="ADRESSE-ELECTRONIQUE" minOccurs="0" />
  </sequence>
</complexType></element>
```

⇒ Choice : choix parmi un ensemble d'éléments (OU logique)

```
<element name="ident"><complexType>
  <choice>
    <element ref="RAISON-SOCIALE" />
    <element ref="IDENTITE" />
  </choice>
</complexType></element>
```

⇒ All : séquence non ordonnée

```
<element name="contac"><complexType>
  <all>
    <element name="NOM.CONTACT" type="identType" />
    <element name="COMMENTAIRE" type="string" minOccurs="0" />
    <element ref="REF.ADRESSE" />
    <element name="DATE.CONTACT" type="date" />
  </all>
</complexType></element>
```

- Utilisable uniquement au niveau le plus haut d'un modèle de contenu et sur des éléments avec maxOccurs à 1 ; pas sur d'autres groupes.

⇒ Contrôle des occurrences : minOccurs, maxOccurs

Instabus XML 2001. Séminaire XML Schémas

4 mai 2001, Pierre-Alain et Bruno Cheval

37

Caractérisation du modèle de contenu

Caractérisation du modèle de contenu

⇒ Mixed Content : inter-relation entre texte et structure

```
<element name="argumentaire-cx">
  <complexType mixed="true">
    <sequence>
      <element name="caract-tech" type="string" />
      <element name="prix" type="string" />
    </sequence>
  </complexType>
</element>
```

⇒ Nillable : le contenu d'un élément peut avoir une valeur nulle

◆ Dans le Schéma

```
<element name="DERNIER.APPEL" type="date"
  nillable="true" />
```

◆ Dans une instance

```
<DERNIER.APPEL xsi:nil="true" /></DERNIER.APPEL>
```

⇒ Déclaration d'un contenu vide

```
<element name="REF.ADRESSE">
  <complexType>
    <attribute name="REF" type="IDREF" />
  </complexType>
</element>
```

Instabus XML 2001. Séminaire XML Schémas

4 mai 2001, Pierre-Alain et Bruno Cheval

38

Attributs

Attributs

- ⇒ Objectif : pouvoir *valuer* des objets structurels
- ⇒ Déclaration dans un élément, un type complexe

- ◆ Attribut

```
<element name="PERSONNE" type="string">  
  <attribute name="GENRE" type="string"/>  
</element>
```

- ◆ Les attributs sont typés (types simples)

```
<attribute name="TEL.TYPE">  
  <simpleType>  
    <restriction base="string">  
      <enumeration value="STANDARD"/>  
      <enumeration value="DIRECT"/>  
      <enumeration value="FAX"/>  
      <enumeration value="MOBILE"/>  
    </restriction>  
  </simpleType>  
</attribute>
```

- ◆ Contraintes

- Contrainte d'optionnalité : use= optional | required | default
- Valeur par défaut : value=string
- Contrainte d'utilisation : prohibited | fixed

Intéressant XSL 2001. Séminaire XSL Schémas

© 2001, Pierre-Henri et Bruno Chérel

30

Documentation et commentaires

- **annotation** : élément pour la documentation
- **documentation** : fils d'annotation
 - Pour être lu par des humains
- **appInfo** : fils d'annotation
 - Information devant être traitée par les applications, feuilles de styles

⇒ Un Schema est un document XML utilisant des espaces de noms
⇒ Rien n'empêche des modèles plus complexes, mixant le modèle et sa documentation, plus organisée

4.2. - Typage des données : définitions de contraintes, de sémantique de données

Utilisation des typages

Utilisation des typages

- ⇒ **Objectifs** : définir des types pour les utiliser dans des modèles de contenus ou des déclarations d'attributs
- ⇒ Un type est identifié et référencé par son nom ou directement défini (type anonyme)

```
<simpleType name="cpType">
  <restriction base="string">
    <pattern value="\d{5}" />
  </restriction>
</simpleType>
<element name="CP" type="cpType"/>
```

Simple

```
<complexType name="caType">
  <sequence>
    <element ref="ADRESSE" minOccurs="1" />
  </sequence>
  <attribute name="DATE-CREATION"
    type="date"/>
  <attribute name="DATE-MAJ" type="date"/>
</complexType>
<element name="carnetAdresse" type="caType"/>
```

Complexe

```
<element name="societe">
  <complexType>
    <sequence>
      <element ref="RAISON-SOCIALE"/>
      <element name="SIRET">
        <simpleType>
          <restriction base="string">
            <pattern value="\d{6}\s\d{3}\s\d{5}" />
          </restriction>
        </simpleType>
      </element>
    </sequence>
  </complexType>
</element>
```

Anonyme

Jinshui X&H, 2001. Séminaire X&H. Schéma

© Mai 2001, Pierre-Alain et Bruno Ch&H

17

Types simples

Types simples

⇒ Permet de sémantiser des contenus textuels

✦ Pour des éléments (nœuds, feuilles) et/ou valeurs d'attributs

✦ Types simples prédéfinis : chaînes de caractères, nombres, temps, XML, ...

- string, byte, unsignedByte, binary, integer, positiveInteger, negativeInteger, nonNegativeInteger, nonPositiveInteger, int, unsignedInt, long, unsignedLong, short, unsignedShort, decimal, float, double, boolean, time, timeInstant, timePeriod, timeDuration, date month, year, century, recurringDay, recurringDate, recurringDuration, Name, QName, NCName, uriReference, language, ID, IDREF, IDREFS, ENTITY, ENTITIES, NOTATION, NMTOKEN, NMTOKENS

⇒ Mécanisme de contraintes


✦ Possibilité de créer de nouveaux types, basés sur des types prédéfinis

✦ Utilisation des facets

- length, minLength, maxLength, pattern, enumeration, maxInclusive, minInclusive, maxExclusive, minExclusive, precision, scale, encoding, period, duration

✦ Plusieurs facets possibles sur un même type

```
<simpleType name="telNumType">
  <restriction base="string">
    <length value="14" />
    <pattern value="\d{2}-\d{2} -\d{2} -\d{2} -\d{2}" />
  </restriction>
</simpleType>
<element name="tel" type="telNumType" />
```



Opérateurs

Opérateurs

⇒ List : permet de structurer un contenu selon une liste de valeurs

```
<simpleType name="ListeDeCpTypeFR">
  <list itemType="cpTypeFR" />
</simpleType>
```

✦ Sur lequel on peut appliquer des facets

```
<simpleType name="quatreCpTypeFR">
  <restriction base="ListeDeCpTypeFR">
    <length value="4" />
  </restriction>
</simpleType>
```

⇒ Union : permet de "fusionner" deux types simples

```
<simpleType name="cpTypeInternational">
  <union memberTypes=" quatreCpTypeFR cpTypeUS" />
</simpleType>
<element name="codePostal" type="cpTypeInternational">
```

⇒ Les instances suivantes sont valides

```
<codePostal>75010 75012 75014 13005</codePostal>
<codePostal>NY541</codePostal>
```

Patterns

Patterns

⇒ Utilisation d'expressions régulières pour contrôler un contenu

- ◆ Utilisables en tant que facets
- ◆ Applicables sur tous les types prédéfinis
 - Sauf : binary, IDREFS, ENTITIES, NMTOKENS
- ◆ Supportent Unicode
- ◆ Proches des langages de Regexp usuels

Chapitre\s\d	trouve	Chapitre 1, Chapitre 4
\p{Lu}	trouve	N'importe quel caractère Majuscule
\p{IsGreek}	trouve	N'importe quel caractère Grec (défini comme bloc Unicode)
\P{IsGreek}	trouve	N'importe quel caractère NON Gr
[^0-9]x	trouve	Tout caractère (non Chiffre) suivi du caractère x
ab{2,4}x	trouve	abbx, abbbx, abbbbx

Intésumé XML 2001. Séminaire XML Schémas

© 2001, Pierre-Alain et Bruno Chézet

40

Types complexes [code 7]

Types complexes



⇒ Définition de modèles de contenus

- † Avec ordonnancement d'éléments-fils
- † Avec définition d'attributs

⇒ Un type complexe contient :

- † Séquence et/ou choix d'éléments-fils
- † Mixed content
- † Attributs typés

⇒ Les types complexes peuvent contenir des définitions de types et des déclarations d'éléments

```
<complexType name="adresseType">
  <choice>
    <element name="ADRESSE-PRO" type="adresseContentType"/>
    <element name="ADRESSE-PERSO" type="adresseContentType"/>
  </choice>
  <attribute name="TYPE">
    <simpleType>
      <restriction base="string">
        <enumeration value="PERSO"/>
        <enumeration value="PROF"/>
      </restriction>
    </simpleType>
  </attribute>
</complexType>
```

Eléments

Attributs

Intésumé XML 2001. Séminaire XML Schémas

© 2001, Pierre-Alain et Bruno Chézet

41

Code 7 : <http://demo-schema/d4-s41/>

4.3. - Concepts avancés

Identification et référencement

Identification et référencement

⇒ Possibilité de définir l'unicité d'objets dans un arbre (sous-arbre) :
Unique

† L'unicité est liée à un champ : un ensemble de nœuds

† L'unicité est liée à un ou plusieurs objets : path par rapport à la racine

```
<unique>  
  <selector>./SOCIETE</selector>  
  <field>SIRET/text()</field>  
</unique>
```

```
<unique>  
  <selector>./AUTEUR</selector>  
  <field>./NOM/text()</field>  
  <field>./PRENOM/text()</field>  
</unique>
```

⇒ Mécanisme de contrôle de liens :
Key/KeyRef

† Pour chaque valeur définie dans KeyRef doit exister une valeur définie par Key

```
<key name="k1">  
  <selector>./ADRESSE</selector>  
  <field>./@ID</field>  
</key>
```

```
<keyRef ref="k1">  
  <selector>./ADR.REF</selector>  
  <field>./@IDREF</field>  
</keyRef>
```

⇒ Généralisation du mécanisme ID/IDREF des DTD
⇒ Unicité par concaténation
⇒ Quelque soit le contenu, l'attribut ou le texte
⇒ Aussi dans un sous-arbre

4.3.1. - Schema et NameSpace

Prise en compte des espaces de noms dans les Schema

Prise en compte des espaces de noms dans les Schémas

- ⇒ Un Schéma est une collection de :
 - ✦ définitions de types
 - ✦ déclarations d'éléments
 - ✦ contraintes
 - ✦ **Appartenant à un espace de nom**

```
<schema
  targetNamespace="http://www.mutu-xml.org/schema/adresse"
  xmlns:ad="http://www.mutu-xml.org/schema/adresse"
  xmlns="http://www.w3.org/2000/10/XMLSchema" . . .
```

- ⇒ Relation Schéma/document
 - ◆ Un document s'appuie sur des espaces de noms
 - Valeur par défaut
 - Plusieurs espaces de noms
xmlns:bcha="http://www.mutu-xml.org/schema/adresse"
 - ◆ Relation de l'instance au Schéma
 - Un Schéma **peut être** identifié dans une instance
 - L'association d'un Schéma à une instance **peut être** réalisée par programme
 - Utilisation de différents Schémas pour une même instance
 - ◆ Possibilité d'association d'un(plusieurs) espace(s) de noms et de son(ses) Schéma(s)

```
<carnet-adresse
  xmlns:ad=http://www.mutu-xml.org/schema/adresse
  xsi:schemaLocation="
    http://www.mutu-xml.org/schema/adresse
    http://www.mutu-xml.org/schema/adresse/adresse.xsd" >
```

Inédit XML 2001. Séminaire XML Schémas © 2001, Pierre-Henri et Bruno Chérel 43

Faire coopérer espaces de noms et Schémas



⇒ Qualification : possibilité d'imposer l'utilisation d'un espace de noms lors d'une "instanciation"

† Sur le Schéma

```
<schema
  targetNamespace="http://www.mutu-xml.org/adresse"
  xmlns:ad="http://www.mutu-xml.org/adresse"
  xmlns="http://www.w3.org/2000/10/XMLSchema"
  elemFormDefault="qualified"
  attributeFormDefault="qualified">
  <element name="personne"/>
  <complexType>
    <element name="nom" type="string"/>
    <element name="prenom" type="string"/>
    <attribute name="age" type="positiveInteger"/>
  </complexType>
</element>
...
```

† Dans l'instance

```
<?xml version="1.0"?>
<iad:personne xmlns:iad="http://www.mutu-xml.org/adresse" iad:age="35">
  <iad:nom>Dupont</iad:nom><iad:prenom>Paul</iad:prenom>
</iad:personne>
```

† S'applique aussi dans le Schéma lui-même

```
<complexType name="nomType"> . . . </complexType>
<element name="nom" type="ad:nomType"/>
```

⇒ Target Namespace non déclaré : pas de qualification

4.3.2. - Factorisation et modularisation

Factorisation

Factorisation

- ⇒ Faciliter l'écriture de modèles de contenus par factorisation
- ⇒ Coopère avec le typage de données

- ◆ Définition

```
<group name="toutesAdresses">
  <sequence>
    <element ref="ADRESSE-POSTALE"/>
    <element ref="ADRESSE-TEL"/>
    <element ref="ADRESSE-ELECTRONIQUE" minOccurs="0"/>
  </sequence>
</group>
```

- ◆ Utilisation

```
<complexType name="adresseContentType">
  <sequence>
    <element name="IDENT" type="identType"/>
    <group ref="toutesAdresses">
  </sequence>
</complexType>
```

- ⇒ Groupe d'attributs : même principe pour les attributs

Modularisation

Modularisation

- ⇒ **Include** : inclusion simple (ce qui est inclus appartient au Schéma local et suit les règles de espaces de noms et qualification)

```
<include schemaLocation="http://www.mutu-xml.org/schema/adresse/nom.xsd"/>
```

- ⇒ **Import** : inclusion "qualifiée" (un espace de nom est associé aux types importés)

```
<schema xmlns:inc=" http://www.mutu-xml.org/schema/adresse/nom/"
targetNamespace=" http:// www.mutu-xml.org/schema/adresse" ...>
  . . .
  <import namespace="http://www.mutu-xml.org/schema/adresse/nom/"
schemaLocation =<http://www.mutu-xml.org/schema/adresse/nom.xsd">
  . . .
  <element name="nomI" type="inc:nomType"/>
  <element ref="inc:nom"/>
```

- Redefine** : permet d'inclure et d'étendre un Schéma

```
<redefine schemaLocation="http://www.mutu-xml.org/schema/adresse/adresse.xsd ">
  <complexType name=" adresseType ">
    <complexContent>
      <extension base=" adresseType ">
        <sequence> <element name="pays" type="string"/> </sequence>
      </extension>
    </complexContent> </complexType>
  </redefine>
```

Intégré à XSL 2001. Séminaire XSL Schémas

© mai 2001, Pierre-Henri et Bruno Chérel

49

4.3.3. - Dérivation

Principes de dérivation de types

Principe de conception hiérarchique de type : notion d'héritage et de dérivation

- Dans la "vraie vie"
 - Une femme est un individu
 - Un homme est un individu
- En modélisant
 - *Individu* défini comme un type
 - Les types *femme* et *homme* sont dérivés du type *individu*

Dériver : surcharger ou réduire

- Ajouter ou contraindre des caractéristiques d'un type
- Définition d'un nouveau type résultant

Méthode de conception des types

- Organisation hiérarchique et méthodique des définitions de type

Instanciation dynamique :

- Dans les instances, utilisation de types interfaces dynamiquement instanciés
 - Dans un Schema, déclaration d'un élément *personne* de type *individu*
 - Dans un document référant ce Schema, un élément *personne* pourra être de type *femme* ou de type *homme*

Mécanisme de dérivation de types

Objectif : contraindre ou élargir le champ de types définis

- S'applique aussi bien sur un type simple que sur un type complexe

Par extension

- Réutilisation d'un type en ajoutant des informations supplémentaires

Par restriction

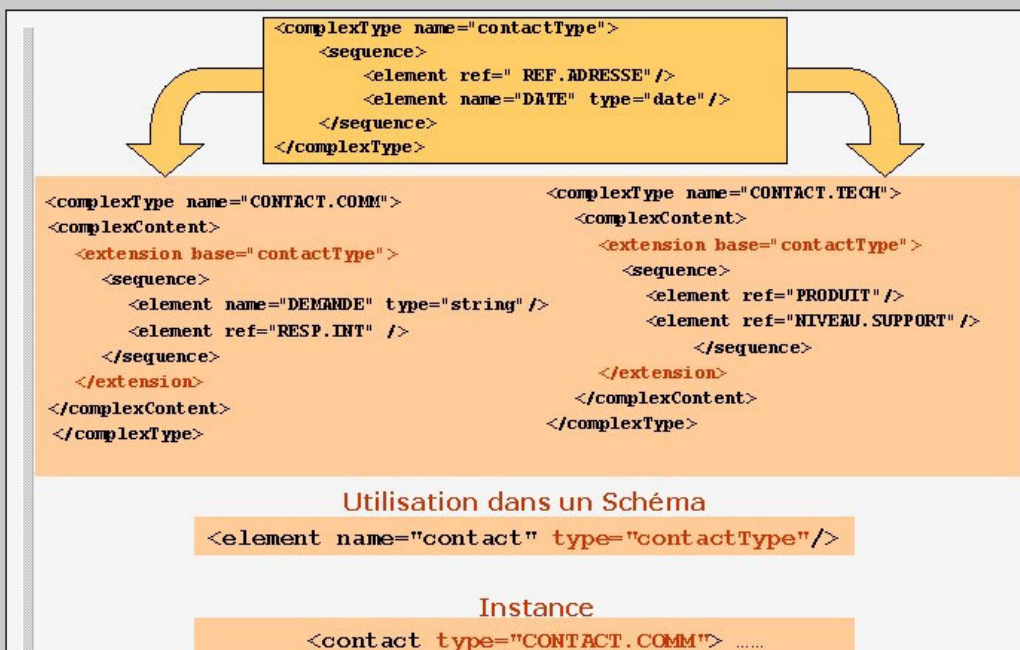
- Réduction du champ de portée d'un type
- Permet d'ajouter des systèmes de contraintes
 - Par exemple sur type complexe, dire qu'un fils est en `minOccurs` à 1
 - Pour un type simple, donner un facet (un `positiveInteger` à un `maxExclusive` à 100)

Possibilité de contraindre la dérivation

- Clarification et maîtriser les architectures de structures
- Utile dans des environnements modulaires de réutilisation
 - `final` : empêche un type d'être dérivé (**extension, restriction, #all**)
 - `fixed` : s'applique sur un facet (dérivation de type simple) pour figer la valeur
 - `block` : empêche l'utilisation d'un type pour dérivation (**extension, restriction, #all**)

Exemple de dérivation par extension

Exemple de dérivation par extension




Jérôme L. 2001, Séminaire XML Schémas

4 mai 2001, Pierre-Alain et Bruno Chéaz

33

Types abstraits et classe d'équivalence



- ⇒ **Objectif** : instanciation dynamique d'éléments et de types
- ⇒ **Types abstraits** : offrent une "interface" pour un élément ou un type à dériver
 - ✦ Nécessitent d'être dérivés

```
<complexType name="PERSONNE" abstract="true" />
<complexType name="HOMME" >
  <complexContent><extension base="PERSONNE" /></complexContent>
</complexType>
<complexType name="FEMME" >
  <complexContent><extension base="PERSONNE" /></complexContent>
</complexType>
<element name="INDIVIDU" type="PERSONNE" />
```

L'élément INDIVIDU pourra avoir le type HOMME ou FEMME dans une instance

- ⇒ **Classe d'équivalence** : permet à des éléments d'être substitués par d'autres dans les instances
 - ✦ Ne relève pas du mécanisme de dérivation
 - ✦ Éléments attachés à une classe d'élément

```
<element name="CONTACT.NOUVEAU" type="contactType"
substitutionGroup="CONTACT" > ... avec son nom et son adresse
<element name="CONTACT.A.ELIMINER" type="contactType"
substitutionGroup="CONTACT" > ... avec son identification
```

Dans l'instance, les éléments CONTACT.NOUVEAU et CONTACT.A.ELIMINER peuvent être utilisés n'importe où à la place de CONTACT

Instalau XSL 2001. Séminaire XSL Schémas © 2001, Pierre-Henri et Bruno Chérel 34

5. - Mise en place des Schema

Pratique des Schema

Chercher le meilleur compromis

- Entre déclarations locales vs globales
- Entre éléments vs Type
- Utilisation de modèle de contenu variable

Espace de noms

- Dans le Schema, quel est le "default namespace"
- Espace de noms dans les instances : cachés ou exposés
- Factorisation et réutilisation basée sur espaces de noms ?

Un Schema étendu ?

- Utilisation d'un même modèle pour de multiples ingénieries
 - De modélisation (bien sûr !)
 - De validation : schématron, trex, etc.
 - De documentation
 - ...

=> Décider d'une méthode de conception, rester cohérent. Voir <http://www.xfront.com> [www.xfront.com]

Code 9 : <http://demo-schema/d6-s54/>

Etat des recommandations

Recommandations du W3C : 2 mai 2001

- XML Schema Part 0: Primer (<http://www.w3.org/TR/xmlschema-0/>) [www.w3.org/TR/xmlschema-0/]
- XML Schema Part 1: Structures (<http://www.w3.org/TR/xmlschema-1/>) [www.w3.org/TR/xmlschema-1/]
- XML Schema Part 2: Datatypes (<http://www.w3.org/TR/xmlschema-2/>) [www.w3.org/TR/xmlschema-2/]

Modification par rapport à la version utilisée ici

- **NameSpace**
 - <http://www.w3.org/2001/XMLSchema>
 - <http://www.w3.org/2001/XMLSchema-instance>
- **Re-écriture de parties des spécifications jugées trop complexes**
- **Changements mineurs**
 - sur les types prédéfinis (timexxx, binaryxxx, booléens..)
 - Attributs : default et fixed
 - `<xs:attribute ref="xml:lang" use="required"/>`
 - `<xs:attribute ref="xml:space" default="preserve"/>`
 - `<xs:attribute name="version" type="xs:decimal" fixed="1.0"/>`
 - Nullable devient nillable, null devient nill
 - ...

Mise à jour des Schema par programme

- <http://www.w3.org/2001/03/webdata/xsu> [www.w3.org/2001/03/webdata/xsu]

Compatibilité DTD et Schema^[code 10]

DTD vers Schema

- **De typage faible à typage fort**
- **Modèle de contenu : des choix à réaliser**
 - Utilisation de type vs utilisation d'éléments
 - DTD : définition globales d'éléments et référencement local
 - Perte des possibilités supplémentaires des Schémas :
 - types complexes
 - déclaration locales
- **Entités paramètre de modèles de contenus ou d'attributs**
 - Modélisations comme groupe

Schema vers DTD

- **De typage fort vers typage faible !**
- **Laxisme des modèles de contenus liés aux déclarations locales différenciées**
- **Support des espaces de noms : comment faire ?**

Coexistence ... manuelle

- **Pas de double génération automatique**
- **Obligation de déconnecter DTD pour repenser en Schema**

Code 10 : <http://.demo-schema/d7-s58/>

Alternative aux Schema

Complémentaire et/ou concurrents !

- **Schematron**
 - « *An XML Structure Validation Language using patterns in Trees* »
 - *Co-constraints* : pattern de contraintes de structure
 - <http://www.ascc.net/xml/resource/schematron/>
 - <http://sourceforge.net/projects/schematron>
- **TREX**
 - *Tree Regular Expressions for XML*
 - <http://www.thaiopensource.com/trex/> [www.thaiopensource.com/trex/]
- **RELAX**
 - *REgular LAnguage for XML*
 - ISO/IEC DTR 22250-1
 - <http://www.xml.gr.jp/relax/> [www.xml.gr.jp/relax/]
- **SOX**
 - « *Schema for Object-Oriented XML 2.0* »
 - Ancêtre W3C XML Schema
 - <http://www.w3.org/TR/NOTE-SOX> [www.w3.org/TR/NOTE-SOX/]

Mais aussi ...

- **XDR**
 - Stratégie d'attente de Microsoft
- **DTD**
 - Appelé à disparaître au profit des Schema, plus puissants
 - *Quid* des entités ?
- **DCD (Document Content Description), DDML (Document Definition Markup Language, DT4DTD (DataTypes for DTDs), DSD (Document Structure Description)...**

=> La veille technologique s'impose

=> On peut commencer à implémenter : rec W3C = stabilité

Outils

- **Parsers : de bons espoirs maintenant que la spécification est stable**
 - XSV : Validation en ligne ou sur poste de travail Windows
 - <http://www.w3.org/2001/03/webdata/xsv> [www.w3.org/2001/03/webdata/xsv]
 - XSV : Upgrade des Schema de la version 20001024 vers l'actuelle (en ligne)
 - <http://www.w3.org/2001/03/webdata/xsv> [www.w3.org/2001/03/webdata/xsv]
 - MSXML
 - V4 : <http://msdn.microsoft.com/xml/general/newinaprilre.asp> [msdn.microsoft.com/xml/general/newinaprilre.asp]
 - OraXml, Xerces, JASP
- **Editeurs**
 - XML Authority, XML Spy v 3.5
- **Conversion DTD : est-ce la bonne pratique ?**
 - DTD2Schema (perl)
 - http://www.w3.org/2000/04/schema_hack/ [www.w3.org/2000/04/schema_hack/]
 - XML Authority (aussi éditeur)
 - <http://www.extensibility.com/> [www.extensibility.com/]
 - XML Spy (aussi éditeur)
 - <http://www.xmlspy.com/> [www.xmlspy.com/]
- **Conformité**
 - La notion de conformité est définie dans la spécification des Schema
 - Groupe de test de la conformité des processeurs
 - SourceForge, <http://xmlconf.sourceforge.net/?selected=schema> [xmlconf.sourceforge.net/?selected=schema]
 - Oasis Schema Conformance Committee, <http://www.oasis-open.org/committees/schema-conformance/index.shtml> [www.oasis-open.org/committees/schema-conformance/index.shtml]

Bilan

Intérêt

- **Conception : modélisation de classe d'objets**
 - Lisibilité
 - Compatibilité XML et Namespace
 - Pouvoir de descriptions des structures et des types
 - Pouvoir d'expression de contraintes
- **Exploitation : validation et utilisation multiples**
 - Coopération : plusieurs Schema sur une même instance (modularité)
 - Application spécifique : une instance validée et utilisée dans un contexte applicatif précis

Limites

- **Forme verbeuse**
- **Contraintes sur les atomes (données), mais pas sur les inter-relations entre les structures**
 - Quel est le rôle du modèle ? Quel est le rôle de l'applicatif ?
 - Complémentarité avec des alternatives
- **Ne permet pas la fragmentation de fichiers de données**
 - Nécessité d'une coopération avec DTD ou utilisation de XInclude

Ecrire un modèle

Une nécessaire concertation sectorielle

- **Luttes d'influence**
 - Écrire un Schema, c'est décider d'un modèle de données
 - un avantage concurrentiel
 - Des luttes sectorielles importantes
 - EbXML vs ICE vs BizTalk vs ...
 - RDF vs XLink vs Topic Maps
- **La bataille des répertoires**

Un Schema d'entreprise

- **Tient compte des impératifs de l'entreprise**
 - Saisie, maintenance, work-flow, assurance qualité
- **Tient compte des besoins d'échanges interpartenaires**
- **Tient compte des échanges avec clients**

Recherche : le "gold" Schema ?

- Solutions intermédiaires : la transformation et la modularisation

Pour en savoir plus sur les techniques

- **Mutu-XML** [www.mutu-xml.org/projet.html]
 - Glossaire et base de liens (articles, tutoriels, etc.)
 - <http://www.mutu-xml.org> [www.mutu-xml.org]
- **OASIS Open, Robin Cover**
 - Recensement de l'activité sur les Schema
 - <http://www.oasis-open.org/cover/schemas.html> [www.oasis-open.org/cover/schemas.html]
- **xFront, Roger L. Costello**
 - Tutoriel de référence
 - <http://www.xfront.com> [www.xfront.com]
- **Best practices**
 - Travail collectif sur les façons de mettre en œuvre les Schema
 - <http://www.xfront.com/BestPractices.html> [www.xfront.com/BestPractices.html]
- **XML-Dev**
 - Liste des développeurs XML
 - <http://lists.xml.org/archives/xml-dev/> [lists.xml.org/archives/xml-dev/]
- **XMLSchema-Dev**
 - Liste des acteurs de la mise en place de la Recommandation
 - <http://lists.w3.org/Archives/Public/xmlschema-dev/> [lists.w3.org/Archives/Public/xmlschema-dev/]