Initiation ODD

Lou Burnard Consulting

A quoi ca sert, un ODD?

Nous souhaitons utiliser XML. Qu'est-ce qu'il nous faut?

- un schéma formel (utilisant un langage informatique tel que DTD, RELAX NG, W3C Schema, Schematron) qui peut contrôler
  - quelles balises sont disponibles ?
  - dans quels contextes ?
  - avec quels attributs ?
  - avec quelles valeur ?
  - en respectant quelles contraintes ?
- Une documentation pour expliciter nos principes éditoriaux, nos principes de choix de balises, etc. aux utilisateurs / développeurs :
  - dans plusieurs langues naturelles ;
  - dans plusieurs formats de fichiers (PDF, Word, HTML, epub...).
- Des outils informatiques pour transformer et gérer nos données.

L'idée essentielle de ODD

One Document Does it all

Un vocabulaire spécialisé pour la définition :

- des schémas
- des types d'élément XML, indépendants des schémas
- des regroupements de tels éléments, publics ou privés
- des patrons (MLE macros)
- des classes (et sous-classes) d'éléments
- des références de tels objets

Un schéma peut combiner :

- des objets identifiables (dans la liste ci-dessus)
- des objets appartenant à d'autres espaces de nom

et devrait être intégrable à un système de balisage documentaire classique

Eléments essentiel d'un ODD

<schemaSpec> définit et identifie un schéma

<elementSpec> fournit une définition d'élément, entièrement ou en partie

<elementRef> utilise une définition d'un élément existant

<classSpec> fournit une définition d'une classe

<classRef> utilise une définition de classe existante

<moduleRef> fournit un ensemble de spécifications d'éléments en faisant référence à un 'module'

(Nous reviendrons plus loin sur les composants documentaire d'un ODD)

Premier exemple simplissime

Nous utilisons un élément <book>, qui contient un mélange des <para>s et de <image>s. Nous ne connaissons rien au sujet de la TEI, et n'en avons pas envie. De même pour les espaces de noms.

```
<schemaSpec ns="" start="book"
  ident="bookSchema">
  <elementSpec ident="book">
    <desc>Elément racine d'un schéma simplissime pour encoder les livres</desc>
    <desc/>
    <content>
      <alternate maxOccurs="unbounded">
        <elementRef key="para"/>
        <elementRef key="image"/>
      </alternate>
    </content>
  </elementSpec>
<!-- ... continue à la prochaine diapo -->
</schemaSpec>
```

exemple simplissime (suite)

```
<schemaSpec ns="" start="stuff"
  ident="oddex-1">
<!-- ... contd -->
  <elementSpec ident="para">
    <desc>un paragraphe de text </desc>
    <content>
      <textNode/>
    </content>
  </elementSpec>
  <elementSpec ident="image">
    <desc>un élément vide qui pointe sur un fichier graphique</desc>
    <content/>
    <attList>
      <attDef ident="href">
        <desc>fournit l' URI de l'objet ciblé</desc>
        <datatype>
          <dataRef name="anyURI"/>
        </datatype>
      </attDef>
    </attList>
  </elementSpec>
</schemaSpec>
```

So what ?

- On peut maintenant générer un schéma RELAX NG, W3C, ou DTD par une transformation XSLT
- On peut extraire les fragments documentaires, notamment les descriptions des éléments et des attributs

TEI fournit un élément spécialisé pour cela :

```
<specList>
  <specDesc key="para"/>
  <specDesc key="picture"/>
</specList>
```

Ce balisage généra quelque chose comme:

<para> un paragraphe de texte

<picture> un élément vide qui pointe sur un fichier graphique

Essayons cela avec oXygen...

D'abord nous allons creer un document ODD et le transformer...

- Démarrez oXygen
- Créez un nouveau fichier (CTRL-N)
- Dans le dialog Nouveau ouvrez le menu Modèles du framework et sélectionnez TEI-ODD -> ODD Customization; cliquez le bouton Créer
- Remplacez la ligne `<schemaSpec ident="myTEI">` proposé en bas par le contenu du fichier oddex-1.xml
- Insérez le contenu du fichier oddex-1-doc.xml *avant* le nouveau <schemaSpec>
- Enregistrez votre ODD sous le nom oddex-1.odd
- Sélectionnez les Scénario de Transformation TEI ODD to RELAX NG XML et TEI ODD to HTML pour générer un schéma et sa documentation à partir de cet ODD

Utilisons notre schema...

- Créez un nouveau fichier (CTRL-N)
- Dans le dialog Nouveau ouvrez le menu Nouveau document et sélectionnez XML Document; cliquez le bouton Personnaliser
- Dans le dialog Nouveau qui s'affiche il faut spécifier l'URL du schema souhaité: vous pouvez le trouver en parcourant les fichiers locaux
- Dans le dialog Ouvrir qui s'affiche naviguez jusqu'au fichier schéma oddex-1.rng que vous venez de créer dans le dossier out et cliquez sur Ouvrir
- Le dialog Nouveau s'affiche encore; mais cette fois avec des informations prises dans votre schéma: cliquez sur le bouton Créer
- Un document conforme à votre schéma est crée. Quels balises rend-t-il disponibles? Quels attributs? Essayez d'ajouter plusieurs <para> et au moins un <image> à votre document.
- Enregistrez votre document par ex avec le nom myDoc-1.xml

Notions de classe 1

Dans le monde réel, le contenu de nos <book> ne se limite pas aux <para>s et <image>s... on peut regrouper tous les éléments qui ont la possibilité d'apparaître au sein d'un book : nous appelons ce regroupement une *classe*, pour laquelle nous proposons le nom bookPart.
Nous utilisons l'élément <classes> pour indiquer l'association d'un élément avec une classe :

```
<elementSpec ident="para">
<!-- ... -->
  <classes>
    <memberOf key="bookPart"/>
  </classes>
<!-- ... -->
</elementSpec>
```

Et voici la définition de la classe bookPart.

```
<classSpec ident="bookPart"
  type="model">
  <desc>éléments qui ont la possibilité de figurer dans un
<gi>book</gi>
  </desc>
</classSpec>
```

Usage d'une classe de modelisation

Maintenant, au lieu de lister exhaustivement tous les composants possibles d'un <book>, il suffit de dire que cet element est composé des membres de la classe bookPart.

```
<elementSpec ident="book">
 <desc>Elément racine d'un schéma simplissime pour encoder
les livres</desc>
 <content>
  <classRef key="bookPart"
   minOccurs="1" maxOccurs="unbounded"/>
 </content>
</elementSpec>
```

(Dès que nous découvrirons l'existence de listes dans les livres nous saurons quoi faire)

Définition d'une classe d' attribut

Dans le monde réel, il est très probable que plusieurs éléments différents comportent les même attributs: il sera donc très pratique de les définir en une seule fois

ODD nous permet de dire que tous les éléments ayant en commun un ensemble d'attributs constituent une classe *attribute class* que nous définissons ainsi

```
<classSpec ident="pointing"
 type="atts">
 <desc>les membres de cette classe ont tous un attribut
<att>href</att>
 </desc>
 <attList>
  <attDef ident="href">
   <desc>fournit l' URI de l'objet ciblé</desc>
   <datatype>
    <dataRef name="anyURI"/>
   </datatype>
  </attDef>
 </attList>
</classSpec>
```

Testez votre compréhension

- Ouvrez le fichier oddex-2.odd avec oXygen et comparez le avec oddex-1.odd
- Créer un nouveau version du schéma à partir de cet ODD
- Est-ce que votre fichier test oddex-1-test.xml reste valide contre cette nouvelle version du schéma ?

Classe d'attribut: un exemple

Les valeurs possibles d'un attribut peuvent être contrôlées de plusieurs manières:

- par référence à un *datatype* (type de donnée) standard externe, par ex `anyURI` or `ID`
- ou en fournissant notre propre liste des valeurs avec l'élément <valList>

Par exemple...

```
<classSpec ident="bookAtts"
  type="atts">
  <desc>attributs applicables aux objets contenus par des <gi>book</gi>
  </desc>
  <attList>
    <attDef ident="xml:id">
      <desc>fournit un identifiant unique pour le composant</desc>
      <datatype>
        <dataRef name="ID"/>
      </datatype>
    </attDef>
    <attDef ident="status">
      <desc>indique le statut de l'élément </desc>
      <valList>
        <valItem ident="red"/>
        <valItem ident="green"/>
        <valItem ident="unknown"/>
      </valList>
    </attDef>
```

Tester votre compréhension...

- Insérez dans votre fichier oddex-2.odd la définition de classe d'attribut qui se trouve dans le fichier oddex-3.xml (vous pouvez l'inserer entre les definitions de classe existantes)
- Ajoutez un <memberOf> dans les <elementSpec>s des éléments qui vont participer à cette classe
- Générez un schéma et assurez-vous que le fichier oddex-1-test.xml reste valide avec cette version du schéma.
- Contrôlez que oXygen vous propose ces nouveaux attributs, et qu'il contraint les valeurs possibles

Qu'est-ce que l'on pourrait vouloir ajouter pour bien documenter son système?

Peut-être ...

- Des gloses, des descriptions en plusieurs langues
- Des exemples d'usage
- des contraintes plus sophistiquées
  - modèles de contenu plus complexes
  - contraintes variables selon le contexte

Et comme tout projet de documentation : versioning, référencements extérieurs, mappings ontologiques ....

Descriptions et gloses

```
<elementSpec ident="para">
 <gloss>paragraph</gloss>
 <desc xml:lang="en">marks paragraphs in prose.</desc>
 <desc xml:lang="fr">marque les paragraphes dans un texte
en prose.</desc>
 <desc xml:lang="es">marca párrafos en prosa.</desc>
 <desc xml:lang="it">indica i paragrafi in prosa</desc>
<!-- ... -->
</elementSpec>
```

Exemples d'usage

Evidemment, si on décrit un schéma XML on va inclure des exemples d'usage en XML. Si la documentation s'exprime également en XML, il faut être astucieux... Il y a trois approches possibles :

- tout cacher avec un "CDATA marked section" (magique hérité de SGML)
- tout échapper en utilisent des "références" (`&lt;` etc)
- utiliser un autre espace de nommage

Seul le dernier vous permet de valider vos exemples : un plus très avantageux

Par exemple

```
<eg><![CDATA[<p>un paragraphe</p> ]]></eg>
```

```
<eg>
 <code lang="XML">&lt;p>un paragraphe&lt;/p></code>
</eg>
```

```
 <egXML
xmlns="http://www.tei-c.org/ns/Examples"> <p>un
paragraphe</p> </egXML>
```

Des contraintes plus sophistiquées

- Les modèles de contenu sont exprimés en ‘pure ODD’, indépendamment du langage de schéma
- Les valeurs d’attribut peuvent être controllés avec <valList> element ou <datatype>, à plusieurs niveaux de généralité
- Les contraintes variables selon le contexte peut être exprimées en utilisant l’élément <constraint>

Nous allons introduire tout cela doucement... !

Et enfin, un mot de la TEI

Admettons enfin que notre <para> n'est pas si loin de l'élément TEI <p>, que notre <image> ressemble beaucoup à l'élément TEI <graphic>, et que notre <book> pourrait être considéré comme un élément TEI . Comment ré-écrire ce schéma pour profiter des définitions TEI existantes ?

```
<schemaSpec start="div"
 ident="testSchema-2" source="tei:1.6.0">
 <elementRef key="div"/>
 <elementRef key="p"/>
 <elementRef key="graphic"/>
 <elementRef key="figure"/>
 <moduleRef key="tei"/>
</schemaSpec>
```

L'élément <moduleRef> nous fournit un ensemble de définitions infrastructurales, notamment pour les classes utilisées partout dans le systeme TEI. A part cela, nous n'avons besoin que de référencer les éléments TEI souhaités avec un <elementRef>.

Création d'un schéma TEI

- Chargez le fichier oddex-tei.odd avec oXygen et comparez le avec les versions précédentes
- Transformez ce fichier en schéma, comme d'habitude.
- Le fichier oddex-tei-test.xml contient une version TEI de notre fichier de test initial : validez-le avec le schéma que vous venez de créer.
- Notez qu'un document TEI *doit* utiliser l'espace de nommage TEI
- Notez également que les concepts TEI et les nôtres ne sont pas forcément identiques (par ex, usage de <graphic>)

Plus tard, nous verrons comment la TEI se sert du système ODD...