Initiation ODD

Lou Burnard Consulting

À quoi ca sert, un ODD?

Nous avons besoin de deux choses complémentaires :

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

OK, mais pourquoi ODD?

Ces attentes pourraient etre satisfaites en plusieurs manières. Les avantages ODD :

- un format XML bien établi, d
- faisant partie intégrale du système TEI
- permettant un traitement fortement integré avec d'autres systèmes TEI
- à la longue terme
- standardisée

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) et des spécifications de donnée (datatype)
- 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 dans un système de balisage documentaire classique

L'idée essentielle de ODD 2

Un ODD peut combiner plusieurs spécifications pour un meme objet

- une qui est 'canonique', referencee dans les Guidelines
- une (ou plusieurs) supplementaires, modifiées en partie ou totalement
- ces versions partielles peuvent etre presentes dans l'ODD ou bien heritées d'un ODD intermediaire

Les règles de combinaison sont explicites, mais un peu complèxes...

Éléments essentiel d'un ODD

<schemaSpec> définit et identifie un schéma

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

<elementRef> utilise une définition d'un élément existant

<classSpec> fournit la 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)

Un premier exemple simplissime

Nous utilisons un élément <book>, qui contient un mélange d'éléments <para>s et <image>s. Nous ne connaissons rien 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>Élément racine d'un schéma simplissime pour encoder les livres</desc>
    <desc/>
    <content>
      <alternate maxOccurs="unbounded">
        <elementRef key="para"/>
        <elementRef key="image"/>
      </alternate>
    </content>
  </elementSpec>
<!-- ... la suite à la prochaine diapo -->
</schemaSpec>
```

Un exemple simplissime (suite)

```
<schemaSpec ns="" start="stuff"
  ident="oddex-1">
<!-- ... suite -->
  <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>
          <rng:data type="anyURI"/>
        </datatype>
      </attDef>
    </attList>
  </elementSpec>
</schemaSpec>
```

So what ?

- On peut maintenant générer un schéma RELAX NG, W3C, ou DTD à l'aide d'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…

- Démarrez oXygen
- Créez un nouveau fichier (CTRL-N)
- Sélectionnez TEI-P5 -> ODD Customization dans le menu Cadre des modèles du dialog Nouveau
- Remplacer l'élément <schemaSpec> proposé 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 a partir de cet ODD
- Enregistrer le schéma obtenu sous le nom oddex-1.rng
- Associez votre schéma avec le fichier test oddex-1-test.xml et validez le fichier

Notions de classe 1

Dans le monde réel, le contenu de nos <book> ne se limite pas uniquement aux <para>s et aux <image>s... on peut regrouper tous les éléments qui peuvent 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 élément est composé des membres de la classe bookPart .

```
<elementSpec ident="book">
 <desc>Élément 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>
    <rng:data type="anyURI"/>
   </datatype>
  </attDef>
 </attList>
/classSpec>
```

Testez votre compréhension

- Ouvrez le fichier oddex-2.xml avec oXygen et comparez le avec oddex-1.odd
- Créez une nouvelle version du schéma à partir de cet ODD
- Assurez vous que le 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) externe, par ex `anyURI` or `ID` (ce sont des datatypes standard, définis par le W3C)
- 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>
        <rng:data type="ID"/>
      </datatype>
    </attDef>
    <attDef ident="status">
      <desc>indique le statut de l'élément </desc>
      <valList>
        <valItem ident="red"/>
        <valItem ident="green"/>
```

Tester votre compréhension…

- Insérez dans votre fichier oddex-2.odd la définition d'une classe d'attribut (il y en a une dans le fichier oddex-3.xml)
- Ajoutez un <memberOf> pour les 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 correctement 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…

Des contraintes plus sophistiquées

- Les modèles de contenu peuvent être exprimés en ‘pure ODD’, indépendamment du langage de schéma
- Les valeurs d’attribut peuvent être controllés avec l’élément <valList> ou <datatype>, à plusieurs niveaux de généralité
- Les contraintes variables selon le contexte peuvent 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. À 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-3.odd avec oXygen et comparez le avec les versions précédentes
- Transformez ce fichier en schéma, comme d'habitude.
- Le fichier oddex-3-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…