Créer une personnalisation TEI

Lou Burnard Consulting

ODD est aussi une langage de personnalisation

On se sert du même système pour spécifier ses choix dans le grand bazar de la TEI et pour spécifier le bazar lui-même.

Un ODD de personnalisation est spécifié par rapport au système complet

- en sélectionnant des modules
- en sélectionnant parmi les objets (éléments, classes, datatypes, macros) fournis par ces modules
- en supprimant quelques uns des attributs fournis par ces objets
- en modifiant ou remplaçant quelques parties de ces spécifications (par ex. les valList)
- éventuellement en ajoutant des spécifications d'objets nouveaux

Le traitement d'un ODD implique la résolution de spécifications multiples pour un même objet

Création d'un personalisation

Comme vous le savez déjà, on utilise l'élément <schemaSpec> pour spécifier un schéma: qu'il soit vierge, ou personnalisé.

- L'attribut @*ident* obligatoire fournit un nom pour le schéma
- L'attribut @*start* indique le ou les noms des élément(s) racine(s) du schéma
- L'attribut @*source* indique l'emplacement de la source TEI dans le cas d'une personnalisation (par ex une version spécifique de TEI P5)
- Les attributs @*docLang* et @*targetLang* permettent la sélection des langues à utiliser pour les descriptions d'éléments et pour les noms d'élément respectivement, en supposant la présence dans cette source des traductions requises

```
<schemaSpec start="TEI"
  ident="testschema" source="tei:1.5.0" docLang="fr">
<!-- déclarations -->
</schemaSpec>
```

Composants d'un schemaSpec

- **moduleRef** : tous les objets définis par le module
- **elementSpec** (etc) : un objet nouveau ou modifié
- **elementRef** (etc) : un objet déjà existant

Sélection par inclusion

Vous pouvez spécifier les éléments que vous souhaitez inclure parmi ceux qui sont proposés par un module :

```
<schemaSpec start="TEI"
 ident="testschema">
 <moduleRef key="tei"/>
 <moduleRef key="header"/>
 <moduleRef key="textstructure"
  include="body div"/>
</schemaSpec>
```

ou (presque) également :

```
<schemaSpec start="TEI"
 ident="testschéma">
 <moduleRef key="tei"/>
 <moduleRef key="header"/>
 <elementRef key="div"/>
 <elementRef key="body"/>
</schemaSpec>
```

Attention ! un module peut définir d'autres choses que des éléments. Les attributs *@include* et *@except* ne s'appliquent qu'au éléments

Selection par exclusion

Vous pouvez spécifier les éléments que vous souhaitez supprimer parmi ceux proposés par un module :

```
<schemaSpec start="TEI"
 ident="testschema">
 <moduleRef key="core"
  except="mentioned quote said"/>
<!-- ... -->
</schemaSpec>
```

ou également :

```
<schemaSpec start="TEI"
 ident="testschema">
 <moduleRef key="core"/>
<!-- ... -->
 <elementSpec ident="mentioned"
  mode="delete"/>
 <elementSpec ident="quote"
  mode="delete"/>
 <elementSpec ident="said"
  mode="delete"/>
</schemaSpec>
```

L'attribut @*mode*

Il peut arriver que votre ODD propose plusieurs déclarations pour un même objet. Ici par ex, il y a une déclaration de l'élément <quote> derivée du module core, et une autre fourni explicitement par un <elementSpec>

L'attribut @*mode* contrôle la résolution de déclarations multiples eventuelles

valeur	effet
add	l'objet sera ajouté s'il n'existe pas encore (autrement, aucun effet)
delete	l'objet sera supprimé, s'il existe (autrement, aucun effet)
change	l'objet sera modifié, en combinant les parties précisées avec les parties existantes
replace	l'objet sera modifié, en ajoutant les parties precisées et en supprimant toutes les parties existantes

Exercice de personnalisation (1)

- Ouvrez le fichier transMin.odd avec oXygen
- Testez votre compréhension de ce fichier
- Utilisez oXygen pour en générer un schéma dans votre langage de schéma préférée
- Créez un nouveau document TEI XML qui utilise ce schéma
- Vérifiez les éléments et les attributs disponibles

Exercice de personnalisation (2)

- On découvre quelques problèmes avec ce schema:
  - on souhaite indiquer la présence des glyphes et symboles non-Unicode
  - on souhaite indiquer la présence des noms de personnes et de lieux
- Ajoutez donc les éléments <g> et <name> à votre ODD
- Regénérez votre schéma et testez si ces elements sont maintenant disponibles.

Vous souhaitez ajouter un element non-TEI? C'est aussi possible...

Exercice de personnalisation (3)

La TEI nous propose plusieurs variétés de noms (<persName>, <placeName>, <orgName> etc.) mais rien de particulier pour les noms botaniques. Dans cet exercice vous allez créer un élément <botName> pour remplir cette lacune.

- Revenez dans votre ODD.
- Ajoutez un element <elementSpec>, avec identifiant botName
- Dans cette spécification, il faut ajouter au moins:
  - un <desc> avec une brève description de l'élément
  - dans <classes> une indication de la classement de l'élément, avec au moins un <memberOf> (nous vous conseillons de faire en sorte que cet élément soit membre des classes att.global et model.phrase)
  - dans <content> une indication du contenu du nouveau élément (par ex `<macroRef key="macro.phraseSeq">`
- Regénérez votre schéma et testez l'effet de vos modifications.

Exercice de personnalisation (4)

Notre personnalisation propose beaucoup d'attributs. Nous pouvons supprimer ceux qui ne nous interessent pas en deux manières:

- on peut supprimer une classe *entièrement* avec un redéclaration spécifiant @*mode*='delete':

```
<classSpec type="atts"
 ident="att.declaring" mode="delete"/>
```

- ou on peut supprimer (ou séléctionner) quelques attributs spécifiques sur un <classRef> :

```
<classRef key="att.global.rendition"
 except="rendition rend"/>
```

ou, avec le même effet:

```
<classRef key="att.global.rendition"
 include="style"/>
```

Exercice de personnalisation (5)

Contrôle des attributs

- Supprimez dans votre schéma les classes suivantes : att.declaring, att.fragmentable, att.edition, att.editLike
- Sélectionnez dans votre schéma l'attribut *@key* de la classe att.canonical; supprimez les attributs *@rendition* et *@rend* de la classe att.global.rendition
- Regénérez votre schéma et testez l'effet de vos modifications...

Vous pouvez également ajouter ou supprimer des attributs, en utilisant un <attList>, qui contient des <attDef>...

Usage de <attList>

- Trouvez le bon endroit au sein du <elementDecl> existant pour ajouter un <attList>
- Insérez un <attDef> pour définir un nouveau attribut genus
- Cet attribut fournit le nom du genus auquel appartient l'objet indiqué: expliquez cela dans son <desc>
- La valeur de cet attribut sera conforme au datatype teidata.word
- Insérez un deuxième <attDef> pour définir un nouveau attribut *@status*
- Cet attribut indique le status du nom botanique, par ex s'il est international, vulgaire, etc.
- La valeur de cet attribut conformera au datatype teidata.enumerated
- Sa définition doit donc etre complétée par un <valList> ...à vous de le définir!

Pour conclure...

- Rappel: ODD fournit des éléments adaptés pour la documentation des langues formelles XML
- Un peu plus de schematron
- Confusion des langues: intégration d'un schéma non-TEI

Rappel : ODD est conçu pour faciliter la documentation systématique des systèmes d'encodage

<code>	morceau de code exprimé en n'importe quel langage formel <code>count = 56;</code>
<att>	nom d'attribut <att>target</att>
<gi>	nom d'élément <gi>table</gi>
<ident>	identifiant ou nom d'un objet en n'importe quel langage formel <ident>$content</ident>
<tag>	balise (spécifique à XML) <tag>ptr target="http://www.bbc.co.uk"/</tag>
<val>	valeur par ex. d'un attribut <val>unknown</val>

Contraintes de données avec Schematron

- Une spécification d'élément peut proposer des contraintes supplémentaires sur son contenu en utilisant un ou plusieurs éléments <constraintSpec>
- Ces règles sont exprimées (typiquement) en utilisant le langage ISO Schematron

```
<elementSpec ident="div"
 module="teistructure" mode="change"
   xmlns:s="http://purl.oclc.org/dsdl/schematron">
 <constraintSpec ident="div"
  scheme="isoschematron">
  <constraint>
   <s:assert test="@type='prose' and .//tei:p">une division
prosaique doit contenir au
        moins un paragraphe</s:assert>
  </constraint>
 </constraintSpec>
</elementSpec>
```

L'élément <constraintSpec>

Il définit une contrainte qui s'applique au sein de l'élément dans lequel il est declaré

- L'attribut @*scheme* spécifie le langage dans lequel s'exprime la contrainte ('isoschematron' par défaut)
- L'attribut @*ident* est obligatoire: il fournit un identifiant unique
- Il rassemble une ou plusieurs <constraint>
- L'élément <constraint> contient (typiquement) un <assert> ou un <report> élément de l'espace de nommage `http://purl.oclc.org/dsdl/schematron`
- L'attribut @*test* fournit une expression XPath vers l'objet à tester.

Fonctionnement des règles Schematron

- Le contenu de l'élément <assert> est affiché si le test est false
- Le contenu de l'élément <report> est affiché si le test est true
- Astuce: plusieurs éléments schematron sont disponibles pour enrichir le texte du message affiché, notamment <name> (context) et <value-of> (valeur)
- Voir `http://www.schematron.com/` pour une description plus détaillée

Un schema RNG intégrant ces règles sera autogénéré si l'on utilise le logiciel oXygen pour traiter son ODD

Applications typiques des regles Schematron

- Contraintes de co-occurrence : ‘si l’attribut X a la valeur A, l’élément qui le porte doit contenir un Y’
- Contraintes arithmetique contextuelles : ‘au sein d’un <titleStmt>, on ne permet qu’un seul <title>’
- Contraintes textuelles : ‘Les caracteres ’ et ‘ ne sont pas permis au sein d’un `<p xml:lang="en">`’
- Contraintes contextuelles : ‘mots en francais (xml:lang=’fr’) ne sont pas permis au sein d’un élément latin (xml:lang=’la’)’
- Intégrité référentielle: ‘un pointer exprimé sous la forme d’une URL et commençant par # doit correspondre à un element ayant un *@xml:id* identique quelque part dans le document’

Un schematron plus complexe (1)

```
<constraintSpec ident="validtarget"
 scheme="isoschematron">
 <constraint>
  <s:rule context="tei:*[@target]">
   <s:let name="results"
    value="for $t in
tokenize(normalize-space(@target),'\s+') return
starts-with($t,'#') and not(id(substring($t,2)))"/>
   <s:report test="some $x in $results satisfies $x">
Erreur: Chaque pointer dans
        "<s:value-of select="@target"/>" doit indiquer un
ID dans ce meme document
        (<s:value-of select="$results"/>)</s:report>
  </s:rule>
 </constraint>
</constraintSpec>
```

Un schematron plus complexe (2)

- `normalize-space(@target)`: supprimer les blancs non-signifiants
- `tokenize(normalize-space(@target),'\s+')`: couper la valeur de l'attribut dans des tokens séparés par des blancs
- `starts-with($t,'#')`: ne considérer que les pointeurs locaux
- `not(id(substring($t,2)))`: y-a-t il un attribut *@xml:id* dont la valeur correspond à la valeur indiquée en sélectionnant ce qui suit son 2ème caractère
- `some $x in $results satisfies $x`: expression XPath permettant la validation d'une séquence de valeurs booléennes (vraies/fausses)

Addition des composants d'une schéma nonTEI

On souhaite utiliser l'élément TEI <formula> pour insérer du contenu exprimé en MathML: il faut donc :

1. inclure les composants du schéma MathML
2. modifier le modèle de contenu de l'élément <formula>
3. générer un schéma qui résout les conflits de nommage

ATTENTION : il y a un élément <list> dans TEI mais également dans MathML !

TEI + MathML : le ODD

```
<schemaSpec ident="tei_math"
 prefix="tei_" start="TEI teiCorpus">
 <moduleRef url="http://www.tei-
c.org/release/xml/tei/custom/schema/relaxng/mathml2-
main.rng"/>
 <moduleRef key="header"/>
 <moduleRef key="core"/>
 <moduleRef key="tei"/>
 <moduleRef key="textstructure"/>
 <moduleRef key="figures"/>
 <elementSpec module="figures"
  ident="formula" mode="change">
  <content>
   <rng:ref name="mathml.math"/>
  </content>
 </elementSpec>
<!-- pour l'instant nous supprimons ce macro -->
 <macroSpec ident="macro.anyXML"
  mode="delete"/>
</schemaSpec>
```

L'attribut *@prefix* nous permet de désambiguïser les identifiants ressortant des schémas différentes

TEI + MathML : le document

Le schéma généré va valider par ex :

```
<p>The relevant inequalities and distributions are
<formula notation="MathML">
  <m:math overflow="scroll">
   <m:mn>0</m:mn>
   <m:mo>.</m:mo>
   <m:mn>0</m:mn>
   <m:mn>1</m:mn>
   <m:mo><</m:mo>
   <m:mi>κ</m:mi>
   <m:mo><</m:mo>
   <m:mn>1</m:mn>
   <m:mn>0</m:mn>
  </m:math>
 </formula>, Vavilov distribution, and ... </p>
```

The relevant inequalities and distributions are $0.01 < \kappa < 10$, Vavilov distribution,