

ODD : sujets avancées

Sebastian Rahtz

fevrier 2015

emacs

Rappel : ODD est un langage conçu pour faciliter la documentation systématique des systèmes de documentation

`<code>` morceau de code exprimé en n'importe quel langage formel

`<code>count += 56;</code>`

`<att>` nom d'attribut

The `<att>target</att>` attribute indicates a URI

`<gi>` nom d'élément

The `<gi>table</gi>` element is for...

`<ident>` identifiant ou nom d'un objet en n'importe quel langage formel

The `<ident>model.biblLike</ident>` class

`<val>` valeur d'attribut

Cet attribut porte la valeur `<val>unknown</val>`

`<tag>` balise XML

```
<ptr target="http://www.bbc.co.uk"/>
```

Dans un ODD il est possible de regrouper des declarations pour faciliter leur exposition:

```
<p>Nous n'utilisons que ces elements du module  
<ident>linking</ident>:  
<specGrp xml:id="linkingElts">  
<!-- quelques elementSpec etc -->  
</specGrp>  
</p>
```

Ensuite, dans le `<schemaSpec>`, on integre ces declarations avec un `<specGrpRef>`

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="teisttructure" 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
prosaïque 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 déclaré

- 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 règles Schematron

- Contraintes de co-occurrence : 'si l'attribut X a la valeur A, l'élément qui le porte doit contenir un Y'
- Contraintes arithmétique contextuelles : 'au sein d'un `<titleStmt>`, on ne permet qu'un seul `<title>`'
- Contraintes textuelles : 'Les caractères ' 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)

Il n'est pas obligatoire d'utiliser l'approche Schematron

On pourrait également écrire de l'XSLT pur et simple pour valider son document, en utilisant `<xsl:message>`.

```
<xsl:template match="q">
  <xsl:if test="count(ancestor-or-self::q)>3">
    <xsl:message>Trois niveaux de quotes? tu es
sûr????</xsl:message>
  </xsl:if>
</xsl:template>
```

Traitement d'un ODD

Un outil ODD doit

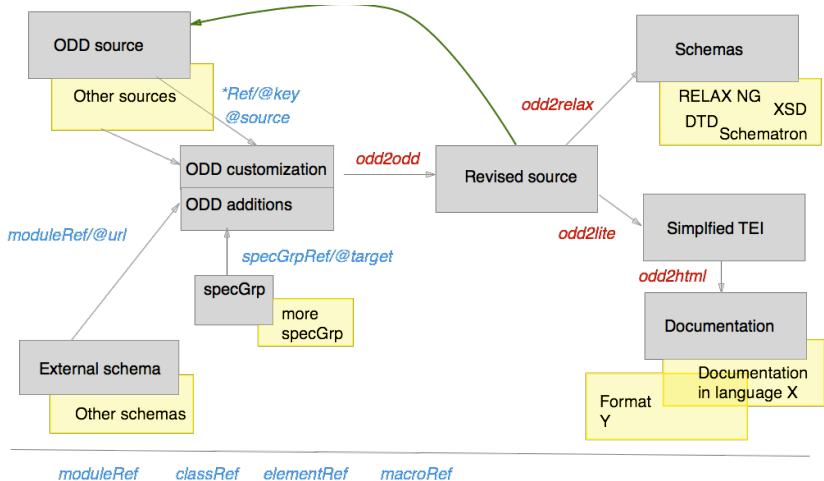
- rassembler les déclarations référencées ou fournies directement
- résoudre les conflits éventuels
- valider la cohérence de ce qui en ressort
- produire un (ou plusieurs) schéma(s) classique(s)
- produire un document XML standard pour documenter ces schémas

<http://www.tei-c.org/Roma/>

<http://tei.it.ox.ac.uk/Byzantium/>

<http://oxgarage.oucs.ox.ac.uk:8080/ege-webclient/>

Organigramme ODD



Usage de l'attribut *@source*

L'attribut *@source* sert à spécifier la source des déclarations qu'on souhaite intégrer

```
<schemaSpec ident="test1"
  prefix="tei_" start="TEI"
  source="http://www.tei-
c.org/Vault/P5/current/xml/tei/odd/p5subset.xml">
<!-- ... -->
</schemaSpec>
<schemaSpec ident="test2"
  prefix="tei_" start="TEI" source="tei:1.5.0">
<!-- ... -->
</schemaSpec>
<schemaSpec ident="test3"
  prefix="tei_" start="TEI"
  source="http://www.tei-
c.org/Vault/P5/1.5.0/xml/tei/odd/p5subset.xml">
<!-- ... -->
</schemaSpec>
<schemaSpec ident="test4"
  source="file:///home/lou/Public/TEI-
SF/P5/Source/Guidelines/en/guidelines-en.xml">
<!-- ... -->
</schemaSpec>
```

Enchainement des ODD

L'opération 'ODD to RELAX NG' comporte deux étapes ('odd to odd' suivi de 'odd to relax') ; pour créer son propre base de personnalisation réutilisable il faut garder le produit de la première étape

- En roma utilisez l'option - - compile
- Utilisez la ligne de commande teitoodd
- Avec OxGarage convertissez du format 'ODD Document' vers 'Compiled ODD Document'
- Avec oXygen générez une nouvelle transformation Ant ; utilisant la feuille XSLT odd2odd.xsl et les options définis par stylesheets/odd/build-to.xml

Addition des composants d'une schéma nonTEI

On souhaite utiliser l'élément TEI `<formula>` et y 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>
</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> $\kappa$ </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,

D'autre ODD outputs

L'outillage actuelle peut faire sortir d'un ODD

des schema DTD, XSD, RELAX NG (XML and compact), ISO Schematron

de la documentation HTML, LaTeX (and thence PDF), XSL FO, Word etc)

et en plus

JSON odds/odd2json.xsl produit un JSON definissant la structure de votre schema, pret a utiliser avec un bibliotheque Javascript

ce qui permet des visualisations...

