

Comment la TEI utilise ODD

Lou Burnard Consulting

TEI ODD sous le capot

La TEI est définie avec des ODD. C'était en effet la raison principale de l'invention du système.

La source TEI P5 (disponible à <http://www.tei-c.org/release/xml/tei/odd/Source/>) rassemble:

- 39 fichiers en TEI-XML, dont 25 contiennent un chapitre de documentation en TEI-XML, la plupart définissant un module, par ex PH-PrimarySources.xml
- 778 fichiers en TEI-XML, chacun définissant un élément, une classe, ou une macro
 - 29 datatypes (`teidata.xxxx.xml`) par ex. `teidata.sex.xml`
 - 116 classes de type modèle (`code>model.xxxx.xml`) par ex. `model.biblLike.xml`
 - 71 classes de type attribut (`att.xxxx.xml`) par ex. `att.divLike.xml`
 - 8 macros (`macro.xxxx`) par ex. `macro.phraseSeq.xml`
 - 555 spécifications d'éléments de `ab.xml` jusqu'à `zone.xml`

Organisation physique des Guidelines

Il ne faut pas confondre l'organisation physique (en fichiers) avec l'organisation logique (en spécifications etc.)

- Le fichier guidelines-XX.xml est *la* source des Guidelines pour la langue XX.
- Mis à part quelques liminaires (un TEI Header, la page de titre, etc.), il contient plusieurs lignes comme ceci :

```
<include
    xmlns="http://www.w3.org/2001/XInclude" href="Guidelines/en/HD-Header.xml"/>
```

- Au sein de chaque chapitre, on trouve des xInclude similaires pour les objets définis par ce chapitre.

Organisation logique des Guidelines

- A la fin de chaque chapitre définissant un module, il y a (par convention) un élément `<moduleSpec>` qui rassemble toutes les spécifications référencées par le chapitre pour définir un module
- Ces spécifications sont organisées (par commodité) en `<specGrp>`, qui sont ensuite référencées par un `<specGrpRef>`
- Chaque `<specGrp>` regroupe des spécifications d'objet, indiqué par un `xlnclude`

Par exemple...

Cette partie de texte des Guidelines:

14.7 Module for Tables, Formulæ, Notated Music, and Graphics

The module described in this chapter provides the following features:

Module figures: Tables, formulæ, notated music, and figures

- Elements defined: [cell](#) [figDesc](#) [figure](#) [formula](#) [notatedMusic](#) [row](#) [table](#)

The selection and combination of modules to form a TEI schema is described in [1.2 Defining a TEI Schema](#).

est généré à partir de ces lignes de code ODD :

```
<div>
  <head>Module for Tables, Formulæ, Notated Music, and Graphics</head>
  <p>The module described in this chapter provides the following features:
    <moduleSpec xml:id="DFTFF" ident="figures">
      <desc>Tables, formulæ, notated music, and figures</desc>
      <desc xml:lang="fr">Tableaux, formules et graphiques</desc>
      <desc xml:lang="zh-TW">表格、方程式與圖表</desc>
      <desc xml:lang="it">Tabelle, formule e figure</desc>
      <desc xml:lang="pt">Tabelas, fórmulas, e figuras</desc>
      <desc xml:lang="ja">図表式モジュール</desc>
    </moduleSpec> The selection and combination of modules to form a TEI
    schema is described in <ptr target="#STIN"/>.
    <specGrpRef target="#DFTTAB"/><specGrpRef target="#DFTFOR"/>
    <specGrpRef target="#DFTNTM"/><specGrpRef target="#DFTGRA"/>
  </p>
</div>
```

Les pointers (#DFTTAB etc.) indique des *specGrp*, comme ceci ...

```
<specGrp xml:id="DFTTAB" n="Tables">
  <include xmlns="http://www.w3.org/2001/XInclude"
    href="../../../Specs/table.xml"/>
  <include xmlns="http://www.w3.org/2001/XInclude"
    href="../../../Specs/row.xml"/>
  <include xmlns="http://www.w3.org/2001/XInclude"
    href="../../../Specs/cell.xml"/>
</specGrp>
```

et le XInclude apporte la specification elle meme d'un *objet*, par ex :

```
<elementSpec module="figures"
  ident="cell">
  <gloss xml:lang="fr"
    versionDate="2007-06-12">cellule</gloss>
  <desc xml:lang="en"
    versionDate="2005-01-14">contains one cell of à table.</desc>
  <!-- ... -->
</elementSpec>
```

De quels types d'objet s'agit-il ?

- datatypes
- model classes
- attribute classes
- macros
- ... et éléments

Nous allons détailler un exemplaire de chacun de ces objets ...

Une spécification ODD

Ouvrez teidata.sex.xml avec oXygen

Comme tout autre spécification TEI...

- C'est un document XML, à valider contre le schéma spécifié
- Avec deux licences open source
- Il y a une description (`<desc>`) répétée en plusieurs langues, chacune identifiée par `@xml:lang` et avec une `@versionDate`
- La spécification est dotée d'un identifiant (`@ident`), et appartient à un module (`@module`)
- Les remarques (`<remarks>`) sont plurilingues de la même manière que les `<desc>` ; noter que les versions françaises et japonaises n'ont pas encore été mises à jour
- Il y a des liens vers des parties des Guidelines où cet objet est présenté, regroupés dans un élément `<listRef>`

Spécification d'un datatype

- L'élément `<content>` fournit la définition du datatype en utilisant un `<dataRef>` qui peut référencier :
 - une spécification TEI en utilisant `@key`
 - une spécification W3C type de base XSD en utilisant `@name`
 - une spécification ailleurs en utilisant `@url`
- Il s'agit ici de la spécification TEI `teidata.word`
- Ce datatype est très répandue en P5: regardons son contenu (ouvrez le fichier `teidata.word.xml`)
- Son `<content>` contient un `<dataRef>` qui fait appel au type de base XSD `token` modifié par un patron
- Ce patron permet n'importe quelle séquence de lettres, chiffres, et caractères de ponctuation sans espaces

Spécification d'une classe modèle

Ouvrez model.biblLike.xml avec oXygen

- La spécification d'une classe modèle n'existe que pour être pointée par d'autres spécifications, donc pas grand chose à voir ici
- Comme ailleurs, on se sert d'un élément `<listRef>` pour regrouper des pointeurs sur la partie des Guidelines où cette classe est décrite.
- Notez l'élément `<classes>` : cette classe est référencée par ("member of") d'autres classes modèles (model.inter, etc.). Une référence à la classe model.inter implique donc une référence à cette classe.
- Pour voir l'effet de cette hiérarchie de classes, regardez la visualisation de cette spécification
<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-model.biblLike.htm>

Spécification d'une classe d'attribut

Ouvrez att.divLike.xml avec oXygen

- On vous recommande d'ouvrir la vue Sommaire (Fenêtre->Afficher la vue->Sommaire) si besoin est pour visualiser la structure
- La liste des attributs fournis par cette classe est spécifiée dans un `<attList>`, qui contient deux `<attDef>` (`@org` et `@uniform`)
- les valeurs disponibles pour un attribut sont spécifiés par un `<datatype>` et éventuellement un `<valList>`, qui rassemblent chaque valeur prévue avec sa définition
- Cette classe est un sous-classe de deux autres (att.metrical, et att.fragmentable) : si ces classes sont disponibles, les attributs qu'elles fournissent seront donc également disponibles.

Spécification d'un élément

Ouvrez abstract.xml avec oXygen

- C'est un élément assez récent (révisé le 2012-12-27) donc il n'y aucune traduction de sa `<desc>` etc. Les parties essentielles d'un `<elementSpec>` sont donc plus évidents.
- Les attributs `@module` et `@ident`
- l'élément `<classes>` qui précise les classes auxquelles cet élément appartient
- l'élément `<content>` qui précise le modèle de contenu de cet élément
- l'élément `<exemplum>` qui contient un exemple de l'usage de l'élément
- facultativement : des remarques (`<remarks>`) et des renvois (`<listRef>`)

Spécification d'une macro

Ouvrez macro.phraseSeq.xml avec oXygen

- Une macro, comme un datatype, fournit une abréviation conventionnelle pour des objets souvent utilisés ; dans ce cas, un modèle de contenu.
- Les modèles de contenu sont exprimés en utilisant des elements ODD :

```
<alternate minOccurs="0"  
  maxOccurs="unbounded">  
  <textNode/>  
  <classRef key="model.gLike"/>  
  <classRef key="model.phrase"/>  
  <classRef key="model.global"/>  
</alternate>
```

Les classes TEI

Si on ne comprend pas le système de classes TEI, on aura du mal à comprendre le fonctionnement d'un ODD

- une classe d'attribut comme `att.global` fournit des attributs
- une classe de modélisation comme `model.profileDescPart` regroupe des éléments selon leur sémantique, ou leur possibilités de positionnement :
 - `model.xxxLike` : ressemble à un `xxx`
 - `model.xxxPart` : constitue un `xxx`
- toute classe peut hériter ses propriétés d'une autre
- on se sert des classes de modélisation surtout dans les définitions de contenu, ce qui permet de les gérer avec une couche d'indirection

Par exemple

```
<classes>
  <memberOf key="att.global"/>
  <memberOf key="att.responsibility"/>
  <memberOf key="model.profileDescPart"/>
</classes>
```

- **<abstract>** est membre de model.profileDescPart
- le contenu de **<profileDesc>** est

```
<classRef key="model.profileDescPart"
  minOccurs="0" maxOccurs="unbounded"/>
```

- en tant que membre de att.responsibility, il hérite des attributs @cert et @resp
- att.responsibility lui transmet également l'attribut @source, parce qu'elle est 'memberOf' de la classe att.source

Définition du contenu d'un élément

- Avant 2014, la TEI se servait du langage RELAX NG pour définir le contenu ("content model") des éléments et des attributs
- (A l'époque, c'était du SGML DTD)
- Depuis 2015, on remplace cela avec des éléments ODD, comme vous l'avez déjà vu
- Pure ODD est plus expressif que DTD ou XSD

Purification d'un modèle de contenu...

```
<content>
  <rng:oneOrMore>
    <rng:choice>
      <rng:ref name="model.pLike"/>
      <rng:ref name="model.listLike"/>
    </rng:choice>
  </rng:oneOrMore>
</content>
```

```
<content>
  <alternate maxOccurs="unbounded">
    <classRef key="model.pLike"/>
    <classRef key="model.listLike"/>
  </alternate>
</content>
```

Voir Burnard et Rahtz 2013 pour les détails

Pure ODD en sommaire

- On précise le type d'objet auquel un modèle fait référence en utilisant `<elementRef>`, `<classRef>`, ou `<macroRef>`
- On utilise les attributs `@minOccurs` et `@maxOccurs` pour indiquer la cardinalité et l'optionalité
- On peut regrouper les références par `<sequence>`, par `<alternate>`, ou par `<interleave>` (mais)
- On utilise un élément spécifique `<textNode>` pour indiquer la présence du texte pur
- un `<content>` vide indique un élément sans contenu
- Un autre élément spécifique `<anyXML>` est proposé

Que signifie la référence à une classe (de modele)?

La signification d'un `<classRef>` au sein de `<content>` varie selon la valeur de ses attributs.

Considérons le cas d'une classe ayant trois membres x, y, z...

| valeur <i>@expand</i> | signification |
|----------------------------|---------------|
| alternate (default) | (x y z) |
| sequence | (x, y, z) |
| sequenceOptional | (x?, y?, z?) |
| sequenceOptionalRepeatable | (x*, y*, z*) |
| sequenceRepeatable | (x+, y+, z+) |

Les attributs *@minOccurs* et *@maxOccurs* sont disponibles, comme ailleurs

Les attributs *@include* et *@except* permettant la sélection ou la suppression de certain membres de la classe également

Exemples

Un exemple vaut un million de mots...

Les expressions du namespace

`http://www.tei-c.org/ns/Examples` sont validées par un schéma spécial, qui permet tout élément TEI comme racine, en utilisant le validateur onvdl.

Contraintes des contenus

Le modèle de contenu n'est que l'une des manières de restreindre ce qui est permis dans un document. On peut aussi exprimer une contrainte ...

- par référence à un élément `<valList>...`
- par reference à un élément `<dataSpec>`
- par inclusion des éléments `<constraintSpec>` (expression en ISO schématron)

Spécification des listes de valeurs

Les valeurs possibles d'un attribut peuvent être spécifiées par un `<datatype>` et/ou par `<valList>`.

Un besoin assez commun est de spécifier une *énumération* (une liste - ouverte ou fermée - de valeurs)

```
<attDef ident="status">
  <desc>indique l'état courant du système selon un code de couleur</desc>
  <defaultVal>green</defaultVal>
  <valList type="closed">
    <valItem ident="red">
      <desc>fermeture complète du système</desc>
    </valItem>
    <valItem ident="orange">
      <desc>fermeture imminente du système</desc>
    </valItem>
    <valItem ident="green">
      <desc>état normal du système</desc>
    </valItem>
    <valItem ident="white">
      <desc>état inconnu du système</desc>
    </valItem>
  </valList>
</attDef>
```

Contraintes schematron

- Une spécification d'élément peut aussi contenir un élément `<constraintSpec>` (ou plusieurs), rassemblant des règles exprimées en ISO schématron
- La TEI s'en sert pour exprimer des contraintes additionnelles non exprimables en DTD etc. par ex des contraintes contextuelles ou concurrentielles
- le traitement de ces règles nécessite une étape additionnelle dans la validation des documents
- mais cela est possible avec la version courante d'oXygen (si votre schéma s'exprime en RELAX NG et si vos contraintes s'expriment en ISO schematron).

Exemple de schématron

Ouvrez span.xml avec oXygen

- cette spécification comporte plusieurs `<constraintSpec>`, chacun avec son `@ident` et son `@scheme`
- un ou plusieurs `<constraint>` sont possibles
- la contrainte s'exprime en langue ISO schématron, donc dans une autre espace de noms
- par ex. `if @to is supplied on <name/>, @from must be supplied as well` (NB "`<name/>`" signifie le nom de l'élément qui activera cette règle)