

# Structuration des données et des documents : balisage XML

## Personnaliser la TEI : *One Document Does it all*

M2 *Humanités numériques*

M2 *Technologies numériques appliquées à l'histoire*

J.B. Camps – 2018



Ce support de cours est mis à disposition selon  
les termes de la licence Creative Commons  
Paternité – Pas d'utilisation commerciale –  
Partage à l'Identique 4.0

# 0. Quelques rappels et approfondissements

# 0.1 Documents valides : les grammaires, DTD et schémas

## - Déclaration DocType (DTD)

syntaxe non XML (héritée de SGML)

## - Schémas

RelaxNG en syntaxe compacte (.rnc) ou XML (.rng)

(W3C) XML Schema

ISO Schematron

# Les DTD (rappels)

`<!ELEMENT nomdelelement (definitionducontenu) >`

N.B. : une seule façon de déclarer un contenu mixte

`<!ELEMENT monélément (#PCDATA | elementenfantoptionnel |  
autrelementenfantoptionnel | etc.)* >`

`<!ATTLIST élément attribut typeDeDonnées #ParDéfaut>`

`<!ENTITY entité "contenu">`

`<!ENTITY documentxmlàinsérer SYSTEM "document/doc.xml">`

`<!ENTITY % nom "contenu" >`

# Attributs

Déclarés par la déclaration `<!ATTLIST >`

`<!ATTLIST élément attribut type #ParDéfaut>`

## Les types de données d'attribut

- CDATA ;
- nom XML (NMTOKEN ou NMTOKENS)
- nom XML unique (ID) ou déclaré ailleurs (IDREF, IDREFS)
- entité (ENTITY ou ENTITIES)
- énumération des valeurs possibles (valeur 1|valeur2|etc.)

# Attributs

Déclarés par la déclaration `<!ATTLIST >`

`<!ATTLIST élément attribut type #ParDéfaut>`

## Les types d'attribut et valeurs par défaut

- `#IMPLIED` : optionnel, pas de valeur par défaut
- `#REQUIRED` : obligatoire, pas de valeur par défaut
- `#FIXED "valeur"` : valeur fixe non modifiable
- "valeur par défaut"

# Les DTD et leurs limitations

## Limitations

- pas de gestion des **espaces de nom** ;
- pas de **typage précis** du contenu des éléments (chaîne de caractères de texte, nombre entier, etc.) ou de leur sens (date, heure, nom, etc.), voire des attributs (en dehors de quelques types très généraux) ;
- peu de **contraintes sur les éléments** (nombre d'occurrences, élément racine,...) ;
- **pas écrites en XML.**  
en revanche : entités, absentes des langages de schéma.

# Les principaux langages de schéma

- (W3C) **XML Schema** (xs ou xsd), dont un des avantages est de permettre un *typage très fin des données* ;
- **Relax NG** (REgular LAnguage for Xml Next Generation) en syntaxe compacte (rnc) ou en syntaxe XML (rng) ; plus simple, souple et très largement utilisé (TEI, Docbook, OpenDocumentFormat, EAD3)
  - la TEI lui est historiquement liée.
- **ISO Schematron**, permet de définir des tests et contraintes de contenu absentes des autres langages.

Voir l'article du Wikipedia anglophone :

« XML Schema Language comparison »

[https://en.wikipedia.org/wiki/XML\\_Schema\\_Language\\_comparison](https://en.wikipedia.org/wiki/XML_Schema_Language_comparison)



# En pratique

Schéma dans **1 seul langage** (par exemple 100 % en XML Schema)

ou **plusieurs** (par exemple, en RelaxNG, avec les types XSD et des contraintes Schematron)

Des conversions (via XSLT) d'un langage de schéma en un autre sont possibles.



# Déclaration d'un élément : comparatif DTD

```
<!ELEMENT nomPropre (#PCDATA) >
```

## XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  <xs:element name="nomPropre" type="xs:string"/>  
</xs:schema>
```

## RelaxNG

### XML

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">  
  <start>  
    <element name="nomPropre">  
      <text/>  
    </element>  
  </start>  
</grammar>
```

## Compact

```
element nomPropre { text }+
```

# Déclaration d'un élément : comparatif

## DTD

```
<!ELEMENT nomPropre (#PCDATA) >
```

## XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mon/espace/de/nom"
attributeFormDefault="qualified">
```

```
  <xs:element name="nomPropre" type="xs:string" >
```

```
    <xs:annotation>
```

```
      <xs:documentation>Cet élément nomPropre sert à encoder les noms
propres, et il ne peut contenir que du texte.</xs:documentation>
```

```
    </xs:annotation>
```

```
  </xs:element>
```

```
</xs:schema>
```

# L'élément **name** dans les *Guidelines*

## Relax NG compact

**element** name

```
{  
  att.global.attributes,  
  att.global.linking.attributes,  
  att.global.analytic.attributes,  
  att.global.facs.attributes,  
  att.global.change.attributes,  
  att.personal.attributes,  
  att.naming.attributes,  
  att.canonical.attributes,  
  att.datable.attributes,  
  att.datable.w3c.attributes,  
  att.datable.iso.attributes,  
  att.datable.custom.attributes,  
  att.editLike.attributes,  
  att.dimensions.attributes,  
  att.ranging.attributes,  
  att.responsibility.attributes,  
  att.source.attributes,  
  att.typed.attributes,  
  macro.phraseSeq
```

```
}
```

## Relax NG XML

```
<rng:element name="name">  
  <rng:ref name="att.global.attributes"/>  
  <rng:ref name="att.global.rendition.attributes"/>  
  <rng:ref name="att.global.linking.attributes"/>  
  <rng:ref name="att.global.analytic.attributes"/>  
  <rng:ref name="att.global.facs.attributes"/>  
  <rng:ref name="att.global.change.attributes"/>  
  <rng:ref name="att.global.responsibility.attributes"/>  
  <rng:ref name="att.global.source.attributes"/>  
  <rng:ref name="att.personal.attributes"/>  
  <rng:ref name="att.naming.attributes"/>  
  <rng:ref name="att.canonical.attributes"/>  
  <rng:ref name="att.datable.attributes"/>  
  <rng:ref name="att.datable.w3c.attributes"/>  
  <rng:ref name="att.datable.iso.attributes"/>  
  <rng:ref name="att.datable.custom.attributes"/>  
  <rng:ref name="att.editLike.attributes"/>  
  <rng:ref name="att.dimensions.attributes"/>  
  <rng:ref name="att.ranging.attributes"/>  
  <rng:ref name="att.typed.attributes"/>  
  <rng:ref name="macro.phraseSeq"/>  
</rng:element>
```

## 0.2 les *Guidelines* et l'infrastructure de la TEI

cf. 1 « The TEI Infrastructure »  
(définie par le module `tei`)

# Les *Guidelines* et leur structure d'ensemble

<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/>

P5: Guidelines for Electronic Text Encoding and Interchange

Version 2.5.0. Last updated on 26th July 2013.

[\[English\]](#) [\[Deutsch\]](#) [\[Español\]](#) [\[Italiano\]](#) [\[Français\]](#) [\[日本語\]](#) [\[한국어\]](#) [\[中文\]](#)



## Front Matter

### [Title](#)

- I. [Releases of the TEI Guidelines](#)
- II. [Dedication](#)
- III. [Preface and Acknowledgments](#)
- IV. [About These Guidelines](#)
- V. [A Gentle Introduction to XML](#)
- VI. [Languages and Character Sets](#)

## Back Matter

- Appendix A [Model Classes](#)
- Appendix B [Attribute Classes](#)
- Appendix C [Elements](#)
- Appendix D [Attributes](#)
- Appendix E [Datatypes and Other Macros](#)
- Appendix F [Bibliography](#)
- Appendix G [Prefatory Notes](#)
- Appendix H [Colophon](#)

## Text Body

- I. [The TEI Infrastructure](#)
- II. [The TEI Header](#)
- III. [Elements Available in All TEI Documents](#)
- IV. [Default Text Structure](#)
- V. [Non-standard Characters and Glyphs](#)
- VI. [Verse](#)
- VII. [Performance Texts](#)
- VIII. [Transcriptions of Speech](#)
- IX. [Dictionaries](#)
- X. [Manuscript Description](#)
- XI. [Representation of Primary Sources](#)
- XII. [Critical Apparatus](#)
- XIII. [Names, Dates, People, and Places](#)
- XIV. [Tables, Formulae, Graphics and Notated Music](#)
- XV. [Language Corpora](#)
- XVI. [Linking, Segmentation, and Alignment](#)
- XVII. [Simple Analytic Mechanisms](#)
- XVIII. [Feature Structures](#)
- XIX. [Graphs, Networks, and Trees](#)
- XX. [Non-hierarchical Structures](#)
- XXI. [Certainty, Precision, and Responsibility](#)
- XXII. [Documentation Elements](#)
- XXIII. [Using the TEI](#)

## TEI sourcecode

- [Using the TEI Sourceforge Repository](#)
- [Sourceforge Subversion Repository](#)
- [Bug Reports, Feature Requests, etc.](#)

# TEI : un modèle d'ensemble et de nombreuses applications

## Distinguer

- **le modèle abstrait de données** (ensemble des concepts et les relations entre ces concepts) et la possibilité d'exprimer des définitions formelles d'une mise en œuvre particulière, un modèle particulier
- les **implémentations techniques du modèle**, (retranscription des contraintes formelles par le biais d'une DTD ou d'un schéma), qui peuvent être
  - générales (correspondre au plus près au modèle des *Guidelines* dans son ensemble)
  - particulières (répondre à des besoins spécifiques, liés à un outil, un projet, etc.)

# La TEI et ses modules

Module name	Formal public identifier	Where defined
analysis	Analysis and Interpretation	<a href="#">17 Simple Analytic Mechanisms</a>
certainty	Certainty and Uncertainty	<a href="#">21 Certainty, Precision, and Responsibility</a>
core	Common Core	<a href="#">3 Elements Available in All TEI Documents</a>
corpus	Metadata for Language Corpora	<a href="#">15 Language Corpora</a>
dictionaries	Print Dictionaries	<a href="#">9 Dictionaries</a>
drama	Performance Texts	<a href="#">7 Performance Texts</a>
figures	Tables, Formulae, Figures	<a href="#">14 Tables, Formulae, Graphics and Notated Music</a>
gaiji	Character and Glyph Documentation	<a href="#">5 Non-standard Characters and Glyphs</a>
header	Common Metadata	<a href="#">2 The TEI Header</a>
iso-fs	Feature Structures	<a href="#">18 Feature Structures</a>
linking	Linking, Segmentation, and Alignment	<a href="#">16 Linking, Segmentation, and Alignment</a>
msdescription	Manuscript Description	<a href="#">10 Manuscript Description</a>
namesdates	Names, Dates, People, and Places	<a href="#">13 Names, Dates, People, and Places</a>
nets	Graphs, Networks, and Trees	<a href="#">19 Graphs, Networks, and Trees</a>
spoken	Transcribed Speech	<a href="#">8 Transcriptions of Speech</a>
tagdocs	Documentation Elements	<a href="#">22 Documentation Elements</a>
tei	TEI Infrastructure	<a href="#">1 The TEI Infrastructure</a>
textcrit	Text Criticism	<a href="#">12 Critical Apparatus</a>
textstructure	Default Text Structure	<a href="#">4 Default Text Structure</a>
transcr	Transcription of Primary Sources	<a href="#">11 Representation of Primary Sources</a>
verse	Verse	<a href="#">6 Verse</a>



# La TEI et ses modules

## **Modules obligatoires, communs à tous les documents TEI**

- **tei** : définition des classes, macros et types de données
- **textstructure** : éléments de base pour structurer un texte de type livre
- **core** : éléments disponibles dans tous les documents TEI
- **header** : en-tête TEI (métadonnées du document)

# Ex., le module `tei`

## « §1.5 The TEI Infrastructure Module

The `tei` module defined by this chapter is a required component of any TEI schema. It provides declarations for all datatypes, and initial declarations for the attribute classes, model classes, and macros used by other modules in the TEI scheme. Its components are listed below in alphabetical order »

# La TEI et ses modules

**Module propres à un type d'objet, une approche, une discipline**

- analysis (analyse linguistique);
- **certainty (niveaux de certitude et marquage de la responsabilité) ;**
- **corpus (corpus) ;**
- drama (textes d'art dramatique) ;
- figures (tableaux, figures et formules) ;
- gaiji (caractères non standard et glyphes) ;
- iso-fs (structures de traits) ;
- linking (liens, segmentation, alignements) ;
- **msdescription (description des manuscrits) ;**
- **namesdates (noms, dates, lieux) ;**
- nets (graphes, réseaux, arbres) ;
- tagdocs (documentation) ;
- **textcrit (apparat critique) ;**
- **transcr (transcription des sources primaires) ;**
- **verse (vers).**

# Ex., le module textcrit

## « 12.5 Module for Critical Apparatus

The module described in this chapter makes available the following components:

Module textcrit: Critical Apparatus

Elements defined: app lacunaEnd lacunaStart  
lem listApp listWit rdg rdgGrp variantEncoding wit  
witDetail witEnd witStart witness

Classes defined: att.rdgPart att.textCritical  
att.witnessed model.rdgLike model.rdgPart »

# Une personnalisation simple de la TEI n'utilisant que les modules (format ODD)

```
<schemaSpec ident="monModele" start="TEI">
```

```
<!-- D'abord, les quatres modules obligatoires-->
```

```
<moduleRef key="header"/>
```

```
<moduleRef key="core"/>
```

```
<moduleRef key="tei"/>
```

```
<moduleRef key="textstructure"/>
```

```
<!-- Ensuite, les modules nécessaires pour les besoins spécifiques de mon  
projet -->
```

```
<moduleRef key="msdescription"/>
```

```
<!-- Car je veux décrire des manuscrits-->
```

```
<moduleRef key="transcr"/>
```

```
<!-- Car je veux aussi les transcrire-->
```

```
<moduleRef key="textcrit"/>
```

```
<!-- Car je veux réaliser un apparat critique -->
```

```
</schemaSpec>
```

# Les classes

<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ST.html#STEC>

**Permettent d'organiser les éléments du modèle. Les éléments héritent des propriétés de la classe de laquelle ils sont membres, mais aussi de la classe de cette classe (*superclass*).**

**Deux types de classes: classes d'attributs (**att.**) et du modèle (**model.**)**

- même ensemble d'attributs (classes d'attributs, **att.**), par ex. [att.naming](#), [att.global](#).
- même emplacement dans le modèle (classes modèle, **model.**), par ex. `model.divPart`, `model.nameLike`

# Classes d'attribut

- même ensemble d'attributs (classes d'attributs, **att.**), par ex.  
classe [att.naming](#) fournit des attributs communs (@role,...) aux éléments qui concernent les personnes, lieux, organismes, etc.  
comme <forename>, <genName>, <nameLink>, <orgName>, <persName>, <roleName>, etc.  
classe **att.global** (@xml:id, etc.), commune à tous les éléments.

Pour prendre un exemple :  
l'arborescence d'**att.naming**



# Classes du modèle

- même emplacement dans le **modèle** (classes modèle, model.)
- classes **structurelles**/hiérarchiques, qui regroupent des éléments qui ont un même emplacement dans la structure hiérarchique d'un document TEI

ex. : **model.divPart** pour ce qui peut être contenu par une div ;  
**model.phrase** pour les éléments pouvant apparaître au niveau de la phrase

- les **classes sémantiques**, qui regroupent les éléments par similarité de sens. Ex. : classe **model.nameLike** pour les divers types de noms; **model.pLike** pour les éléments de type paragraphe ; **model.rdgLike** pour définir des éléments sémantiquement équivalents à un rdg (une leçon/variante)

Pour prendre un exemple :  
**model.respLike**

# Les macros

Les macros sont des raccourcis qui permettent de **spécifier le contenu permis** (contrainte de contenu) :

- comme contenu d'un élément (*Standard Content Models*)  
ex. mélange de texte (PCDATA) et de tels et tels sous éléments, répétés tant de fois, ou dans tel ordre
- comme valeur d'un attribut (*Datatype Specifications*)  
ex. une chaîne de caractère, ou un nom XML, ou une liste séparée par des espaces, etc.

# Les 6 macros de contenu d'élément (*Standard Content Models*) les plus courantes (par ordre décroissant) macro.

**macro.phraseSeq** (*phrase sequence*) : mélange de PCDATA (de texte) et d'éléments de niveau phrase (0 ou plus et dans n'importe quel ordre)

**macro.paraContent** (*paragraph content*) contenu d'élément de type paragraphe

**macro.specialPara** (*'special' paragraph content*) contenu de paragraphes spéciaux, qui peuvent contenir soit du contenu structuré de type sous-éléments, soit du contenu type paragraphe

**macro.phraseSeq.limited** (*limited phrase sequence*) mélange de texte et de contenu de niveau phrase, *excluant les éléments servant à la transcription de sources*

**macro.limitedContent** (*paragraph content*) la même chose, mais au niveau paragraphe

**macro.xtext** (*extended text*) mélange de PCDATA et d'éléments gaiji

Pour prendre un exemple : **macro.phraseSeq**

# Macros de valeur d'attributs

(cf. *Guidelines*, 1.4.2 « Datatype Macros » pour la liste complète)

NB : *data.* devient *teidata.*

## Généralistes

**teidata.word** un mot

**teidata.text** une chaîne de caractères

**teidata.name** un nom XML

**teidata.enumerated** une valeur (nom XML) parmi une liste fermée de valeurs possibles

...

## Propres au modèle de la TEI

**teidata.certainty** degré de certitude (`data.certainty = "high" | "medium" | "low" | "unknown"`)

**teidata.probability** probabilité

**teidata.numeric** valeur numérique

**teidata.interval** intervalle

**teidata.percentage** pourcentage

**teidata.count** compteur

...

# Macros de valeur d'attributs

(cf. *Guidelines*, 1.4.2 « Datatype Macros » pour la liste complète)

## Valeurs normalisées

**teidata.duration.w3c** durée exprimée selon la spécification W3C

**teidata.duration.iso** durée exprimée selon la spécification ISO 8601

...

## Valeurs spécialisées

**teidata.namespace** espace de nom

**teidata.pattern** une expression régulière

**teidata.pointer** pointeur

**teidata.version** une version de la TEI ou d'Unicode

**teidata.versionNumber** un numéro de version

**teidata.xpath** une expression XPath

...

## 0.3 Créer une personnalisation de la TEI



# Modélisation XML, quelques principes généraux

- Un modèle est par nature une **simplification** de la complexité du réel, pour en faciliter l'analyse en sélectionnant des **faits** et en les **formalisant** ;
- Un modèle XML est toujours conçu pour **répondre à un besoin** donné de gestion de documents ou de données (production contrôlée, stockage ou échange d'informations, développement d'applications et de services) .
- La définition d'un modèle de document est une **étape cruciale**, qui conditionne grandement le déroulement d'un projet et ses résultats

Cf. Florence Clavaud, *Modélisation XML : principes*, ENC, 2013.

# Modélisation XML, quelques principes généraux (2)

- Le modèle est le **reflet d'une analyse** de documents : de leur structure, du contenu ; **et d'une sélection** d'informations à modéliser, des besoins d'un projet,...
- Un modèle doit viser à l'**absence d'ambiguïté** et à être **explicite** (pour la machine et la personne)
- Un modèle XML tend à être **essentiellement sémantique**

Cf. Florence Clavaud, *Modélisation XML : principes*, ENC, 2013.

# Lignes directrices pour un bon modèle XML

- définir strictement le contenu des documents, pour guider le travail d'encodage et faciliter les traitements ultérieurs et l'homogénéité des documents
- définir des éléments en adéquation avec les informations portées par les documents
- utiliser des noms explicites et clairs pour l'humain
- regrouper dans un même élément parent les éléments liés entre eux sémantiquement ou qui seront traités de pair
- adopter une répartition cohérente des informations entre éléments et attributs
- être documenté
- être éprouvé en gardant la capacité d'évoluer



# La notion de « TEI Conformance » : objectifs

Est : Un ensemble de contraintes devant permettre

- **l'échange** de fichiers dans un cadre commun
- le traitement par des **outils** partagés
- la fourniture d'un modèle **documenté**

N'est pas : une mesure des mérites académiques  
d'un document



# La notion de « TEI Conformance » : définition

cf. *Guidelines*, 23.4 « Conformance »

« A document is TEI-conformant if it:

- is a well-formed XML document
- can be validated against a TEI Schema, that is, a schema derived from the TEI Guidelines
- conforms to the TEI Abstract Model
- uses the TEI Namespace (and other namespaces where relevant) correctly
- is documented by means of a TEI-conformant ODD file »



# La notion de « TEI Conformance » : en bref

## Deux principes cruciaux à la conformité à la TEI

- ne pas altérer le sémantisme d'éléments de la TEI
- ne peut pas être plus large que l'ensemble de la TEI, créer des possibilités qui n'y existent pas (doit être équivalent ou plus restrictif)  
(ex. **respStmt**)

cf. Sebastian Rahtz & Lou Burnard, « Reviewing the TEI ODD system »

# Typologie brève des modifications

1. suppression d'éléments
2. *changement de nom d'éléments*
3. modification d'un modèle de contenu
4. modification d'une liste d'attributs ou de valeurs d'attributs ; d'un type de données
5. modification de l'appartenance à une classe
6. ajout de nouveaux éléments

En *italiques*, les types de modification intrinsèquement sales.

# Propre ou sale ?

cf. *Guidelines*, 23.3 « **Personalization and Customization** »

Une **modification propre** (« clean modification ») est, dans le jargon de la TEI, une modification qui n'empêche pas la validité d'un document lorsqu'il est confronté aux modules de la TEI dans leur forme originelle

Une **modification sale** (« unclean modification ») est une modification qui ne permet pas cette validité

ex. **att.textCritical**

N.B. : les « modifications sales » ne sont pas intrinsèquement mauvaises, et peuvent même parfois être nécessaires



# Typologie brève des modifications

1. suppression d'éléments
2. *changement de nom d'éléments*
3. modification d'un modèle de contenu
4. modification d'une liste d'attributs ou de valeurs d'attributs ; d'un type de données
5. modification de l'appartenance à une classe
6. ajout de nouveaux éléments

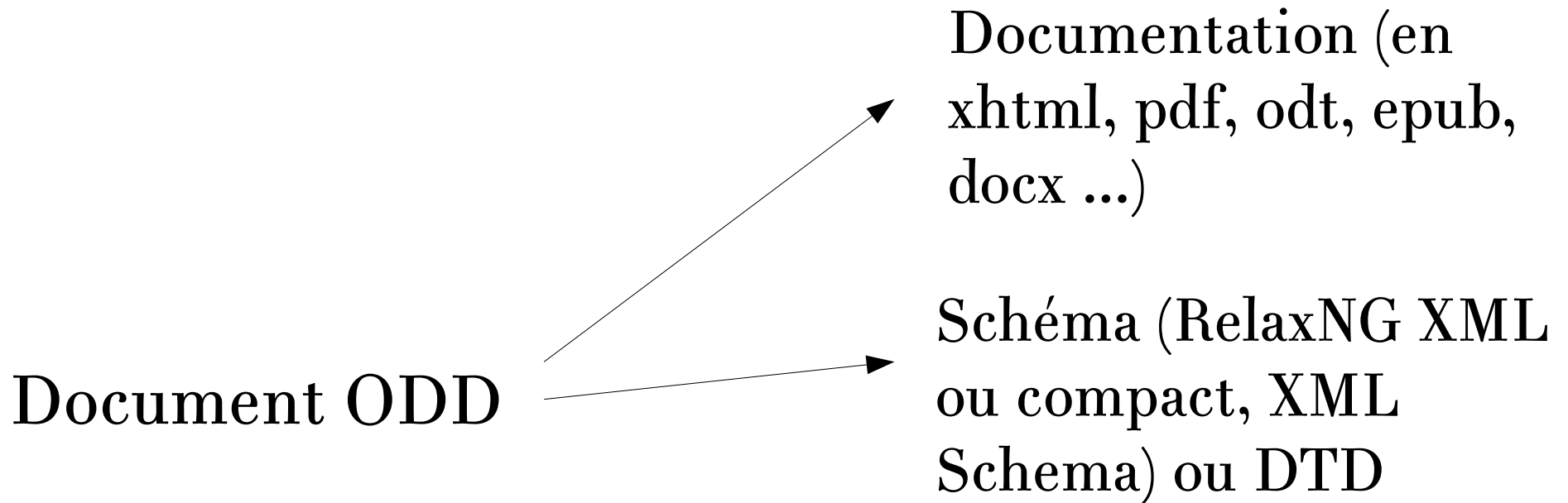
En *italiques*, les types de modification intrinsèquement sales.

# 1. *One Document Does it all*

# 1.1 Principes

cf. *Guidelines*, 22 « Documentation Elements » ;  
23.5 « Implementation of an ODD System »  
(module tagdocs)

ainsi que « Getting Started with P5 ODDs »



# ODD et le concept de « Literate programming »

Donald E. Knuth, *Literate Programming*, CSLI  
Lecture Notes 27, Stanford : Center for the Study of  
Language and Information, 1992.

cf. Sebastian Rahtz & Lou Burnard, « Reviewing the  
TEI ODD system »,

# Créer un *ODD*

## Trois manières principales :

1. Créer un modèle via *Roma* ou assimilé, l'exporter en ODD (et le retravailler) ;
2. Utiliser les feuilles de style *oddbysample* pour créer un fichier ODD (et le retravailler) ;
3. *ex nihilo*.

## 1.2 PizzaChef, Roma, Carthage et Byzantium : des délices culinaires aux raffinements byzantins

PizzaChef : <http://www.tei-c.org/Vault/P4/pizza.html>

Carthage : <https://github.com/TEIC/Carthage>

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

Byzantium (alias *Nova Roma* sive *Roma Secunda*) :  
<http://tei.oucs.ox.ac.uk/Byzantium/> (v. 0.4)

et ensuite ? Moscua ? (*Tertia Roma*)

# Tutoriel Roma :


[http://www.tei-c.org/Guidelines/Customization/use\\_\\_roma.xml](http://www.tei-c.org/Guidelines/Customization/use__roma.xml)



# 1.3 *oddbyeexample*

# Installation de l'add-on TEI à jour sous <oxygen/>

- ✓ Options/Préférences/Association de type de document  
=> tout désactiver
- ✓ Options/Préférences/Add-ons, ajouter  
<http://www.tei-c.org/release/oxygen/updateSite.oxygen>  
appliquer et accepter
- ✓ Aide/Gérer les Add-ons/Installer, installer
- ✓ Lorsque l'installation est terminée :  
Options/Préférences/Association de type de document  
=> tout activer, appliquer, accepter



# Utilisation des feuilles du plugin TEI d'<oxygen/>

- ✓ Vérifier que l'association est bien activée :  
Options/Préférences/Association de type de document => tout activer, appliquer, accepter
- Installer le scénario oddbyexample (pour générer un document ODD à partir d'une collection de documents TEI)**
- ✓ Configurer les Scénarios de Transformation (CTRL+MAJ+C ou menu Document/Transformation/Configurer...), Nouveau, *XML transformation with XSLT*
- ✓ Lui donner un nom et sélectionner « options globales »
- ✓ XSL URL `${frameworks}/tei/xml/tei/stylesheet/tools/oddbyexample.xsl`
- ✓ Sélectionner processeur Saxon 9.x, et aller dans ses options avancées, pour entrer dans template (-it) : main
- ✓ Paramètres : corpus `${cfdu}` (i.e. répertoire courant)
- ✓ éventuellement, configurer la sortie (onglet Sortie)

# 1.4 La Syntaxe ODD

# Arborescence d'ensemble

Un document ODD est un document TEI, contenant la documentation d'un modèle et, quelque part, un élément `schemaSpec`, qui sert de racine aux *spécifications techniques*.

## 1.4.1 Spécifications techniques

**Racine** : élément `<schemaSpec/>`

**Des sous-éléments en :**

*xRef* : référence à une définition préexistante (model.oddRef), que l'on utilise

*xSpec* : (re)définition/spécification d'un objet (model.oddDecl), que l'on fournit

`<moduleRef/>` : référence à un module

`<moduleSpec/>` : définition d'un module

`<elementRef/>` : référence à un élément existant

`<elementSpec/>` : définition d'un élément

`<classRef/>` : référence à une classe et `<classSpec/>` : définition d'une classe

`<macroRef/>` : référence à une macro et `<macroSpec/>`

`<dataRef/>` : référence à un type de données et `<dataSpec/>`

**NB : en pratique, seuls ceux en  
gras vont nous servir.**

# Les attributs des éléments de modèle

**att.identified** : @ident, @module, @predeclare, att.combinable

**@ident** : permet de fournir l'identifiant (du module, de l'élément, de l'attribut... par ex. 'msDescription' ou 'app' ou 'type')

**@module** : le nom du module auquel se rattache l'élément

**@predeclare**\*

+ hérite de **att.combinable** (@mode), voir diapo suivante,  
qui hérite lui-même de **att.deprecated** (@validUntil)

**N.B. :** Si vous vous demandez à quoi sert @predeclare, dont la valeur par défaut est false, mais qui est toujours utilisé dans les *Guidelines* avec la valeur 'true' et seulement sur certaines rares définitions de classes, voir cet [échange sur la liste du TEI-Council](#) entre Martin Holmes et Sebastian Rahtz :

On 23 May 2012, at 16:48, Martin Holmes wrote:

> When I come to look at @predeclare, I find I don't really understand the  
> purpose of it.

few are chosen for the journey to the dark lands, fewer return  
(...)

Pour faire bref, cela ne concerne que le cas où l'on cherche à générer une DTD à partir du fichier ODD et sert à suppléer à une défaillance des processeurs ODD. Lors de la création de la DTD les macros sont transformées en entités, or, certaines macros ont besoin d'être déclarées dans la DTD avant les autres, c'est donc à ça (et à ça seulement) que sert predeclare... en d'autres termes, pour le non averti, « *timey wimey wibbly wobbly... stuff* »



# Les modes (@mode)

'add' : ajout d'un objet nouveau

ex.

```
<elementSpec ident="nouvelElement" mode="add"/>
```

'replace' : substitution de l'objet défini à l'objet de même nom (@ident)

'delete' : supprime toute référence à l'objet du même nom (@ident)

ex. `<elementSpec ident="name" mode="delete"/>`

'change' : modification du contenu de l'objet

```
<elementSpec ident="p" mode="change"> ... </elementSpec>
```

# Définir un module : l'exemple de textcrit dans les *Guidelines* de la TEI

```
<moduleSpec xml:id="DTC" ident="textcrit">  
  <altIdent type="FPI">Text Criticism</altIdent>  
  <desc>Critical Apparatus</desc>  
  <desc xml:lang="fr">Apparat critique</desc>  
  <desc xml:lang="zh-TW">學術編輯註解 </desc>  
  <desc xml:lang="it">Apparato critico</desc>  
  <desc xml:lang="pt">Critical Apparatus</desc>  
  <desc xml:lang="ja">校勘モジュール </desc>  
</moduleSpec>
```

# Définir un module : les premiers modules de la MEI

```
<!-- MODULES -->
  <moduleSpec ident="MEI">
    <desc>Data type definitions.</desc>
  </moduleSpec>
  <moduleSpec ident="MEI.analysis">
    <desc>Analytical component declarations.</desc>
  </moduleSpec>
  <moduleSpec ident="MEI.cmn">
    <desc>Common Music Notation (CMN) repertoire
component declarations.</desc>
  </moduleSpec>
```

# Importer un module : `moduleRef`

## Utilisation des attributs

`@key` : identifiant du module

*et*

`@except` (tout le module sauf...)

*ou*

`@include` (seulement...)

# Quelques exemples...

<moduleRef/>

<moduleRef key="analysis" include="w"/>

Inclut le module analysis mais en n'en reprenant que  
l'élément <w/>

<moduleSpec/>

<moduleSpec ident="namesdates">

<altIdent type="FPI">Names and Dates</altIdent>

<desc>Additional elements for names and  
dates</desc>

</moduleSpec>

Crée un nouveau module, namesdates, alias Names and Dates  
qui est destiné à fournir des éléments supplémentaires pour les  
noms et les dates.

# <elementSpec/>

<gloss/> ou <desc/> : pour le décrire

<classes/> : ses classes

<content/> : son contenu possible

<constraintSpec/> : des contraintes  
supplémentaires

<attList/> : ses attributs

<exemplum/> : des exemples

<remarks/> : des remarques additionnelles

<listRef/> : des références à la documentation

# <elementSpec/>

```
<elementSpec module="tagdocs" ident="code">
  <gloss/>
  <desc>contains literal code</desc> <!--Description de
l'élément -->
  <classes> <!--Liste des classes dont il est membre -->
    <memberOf key="model.emphLike"/>
  </classes>
  <content> <!--Définition de son contenu -->
    <textNode/>
  </content>
  <attList> <!--Liste de ses attributs -->
    <attDef ident="type" usage="opt">
      <desc>the language of the code</desc>
      <datatype>
        <dataRef key="teidata.word"/>
      </datatype>
    </attDef>
  </attList>
</elementSpec>
```

# Les classes

Définir l'appartenance d'un objet à une classe :

À l'intérieur d'un élément `<classes/>`, des sous-éléments `<memberOf/>` utilisant `@key` pour pointer vers les classes dont l'objet est membre.

Par exemple,

```
<elementSpec module="tagdocs" ident="code">  
  <!--[...] -->  
  <classes>  
    <memberOf key="model.emphLike"/>  
  </classes>  
  <!--[...] -->  
</elementSpec>
```

La classe doit être définie par ailleurs (avec un élément `<classSpec/>`).



# Modèles de contenu et conditions (1/3)

Le contenu possible pour un élément peut être défini par un élément `<content/>` ; il peut être spécifié soit par recours à RelaxNG (ODD ancien style), soit en syntaxe nativement TEI (Pure ODD).

Par exemple (Pure Odd),

```
<elementSpec module="tagdocs" ident="code">
  <!--[...] -->
  <content>
    <!--Définition de son contenu en utilisant rng -->
    <!--<rng:text/>-->
    <!-- en Pure ODD -->
    <textNode/>
  </content>
  <!--[...] -->
</elementSpec>
```

# <attList/> et <attDef/>

<attList/> : contient la liste des attributs

<attDef/> : contient la définition d'un attribut

@usage = 'opt', 'rec', 'req'

<valList/> et <valItem/> pour définir une liste de valeurs ('open', 'semi' ou 'closed').

```
<attList> <!--Liste d'attributs pour un élément -->
  <attDef ident="n" mode="delete"/> <!--On suppr. @n-->
  <attDef ident="type" mode="change" usage="req"> <!--On
modifie (change) l'attribut type en lui ajoutant (add) une liste
de valeurs fermée et en le rendant obligatoire-->
    <valList mode="add" type="closed">
      <valItem ident="linguistic"/>
      <valItem ident="semantic"/>
      <valItem ident="lacuna"/>
      <valItem ident="omission"/>
      <valItem ident="translation"/>
    </valList>
  </attDef>
</attList>
```

# C'est mieux en documentant

```
<attList>
  <attDef ident="n" mode="delete" />
  <attDef ident="type" mode="change">
    <gloss>type de variante</gloss>
    <desc>permet la catégorisation des
variantes selon une liste de valeurs close.</desc>
    <valList mode="add" type="closed">
      <valItem ident="linguistic">
        <gloss>d'ordre linguistique</gloss>
        <desc>concerne les variantes d'ordre
linguistique (i.e. graphique, flexionnel, ...).</desc>
      </valItem>
      <!--etc.-->
    </valList>
  </attDef>
</attList>
```

# Modèles de contenu et conditions (2/3)

Il est possible d'utiliser la syntaxe TEI : `<elementRef/>`, `<classRef/>` et `<macroRef/>` peuvent être utilisés pour définir ce contenu, éventuellement en spécifiant des séquences ou un mélange (avec `<sequence/>` et `<alternate/>`), et en utilisant les attributs de la classe `att.repeatable` pour ajouter des contraintes simples :

## **att.repeatable**

@minOccurs : nombre minimal d'occurrences

@maxOccurs : nombre maximal d'occurrences

# Modèles de contenu et conditions (2/3)

## Exemples de la **macro.xtext**

Autoriser tout contenu qui se rattache au modèle  
model.gLike, entre 0 et  $\infty$ , avec éventuellement du  
texte

```
<content>
```

```
  <alternate minOccurs="0"
```

```
    maxOccurs="unbounded">
```

```
    <textNode/>
```

```
    <classRef key="model.gLike"/>
```

```
  </alternate>
```

```
</content>
```

## Modèles de contenu et conditions (2/3)

Modifie l'élément **app** ('change'), pour que son contenu soit constitué dans cet ordre (@preserveOrder) et sans contenu mixte, d'un et un seul élément lem (obligatoire) suivi d'au moins un élément **rdg**, et de **witDetail** optionnels.

```
<elementSpec ident="app" mode="change"> <!--On modifie la déf. d'app -->
  <content>
    <sequence preserveOrder="true"> <!--Les éléments doivent apparaître
dans cet ordre, sans mélange avec des nœuds de texte -->
      <elementRef key="lem" minOccurs="1" maxOccurs="1"/>
<!-- Un et un seul élément lem obligatoire -->
      <elementRef key="rdg" minOccurs="1"
maxOccurs="unbounded"/>
<!-- Au moins un élément rdg obligatoire-->
      <elementRef key="witDetail" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
  </content>
</elementSpec>
```

# Modèles de contenu et conditions (3/3)

Des contraintes de contenu additionnelles peuvent être formulées en utilisant l'élément `<constraintSpec/>`, contenant une contrainte dans un langage formel (par ex. en Schematron).

L'exemple suivant (tiré des *Guidelines*) utilise Schematron pour contraindre la présence d'un élément title possédant un attribut @type de valeur "main" dans le teiHeader :

```
<constraintSpec ident="maintitle" scheme="isoschematron">  
  <constraint>  
    <s:assert  
      test="tei:fileDesc/tei:titleStmt/tei:title[@type='main']"> a  
main title must be supplied  
    </s:assert>  
  </constraint>  
</constraintSpec>
```

# Modèles de contenu et conditions(3/3)

```
<constraintSpec scheme="isoschematron" ident="notice-  
full" mode="add">  
  <constraint>  
    <s:assert test="(@type[. = 'short'] and  
child::tei:p) or (@type[. = 'full'] and  
not(child::tei:p))">Les notices courtes contiennent un  
paragraphe et les notices complètes utilisent les  
éléments structurés.</s:assert>  
  </constraint>  
</constraintSpec>
```



# Modifier une classe

```
<classSpec type="(atts|model)" ident="" module=""  
           mode="">
```

# Macros et types de données

<macroSpec/>

et

<dataSpec/>

## 1.4.2 Documenter

# Documentation : structure de la documentation d'objets

<**gloss**/> : une définition

<**desc**/> (description) : une description courte, commençant par un verbe et contenant une seule phrase.

<**equiv**/> (equivalent) : spécification d'un équivalent

<**name**/> : le nom du concept défini

<**altIdent**/> (alternate identifier) : autre identifiant

<**listRef**/> (list of references) : liste de références

<**remarks**/> : remarques

<**exemplum**/> : un groupe d'exemples

# Et des éléments pour écrire sa prose de documentation : éléments généraux

<**code**/> : pour intégrer du code

<**ident**/> (*identifier*) : contient un identifiant

<**att**/> : contient le nom d'un attribut (forme spécifique d'identifiant)

<**gi**/> : contient le nom (*generic identifier*) d'un élément

<**tag**/> : contient le texte d'une balise ouvrante ou fermante

<**val**/> : contient une valeur d'attribut

<**eg**/> : contient un exemple

<**egXML**/> : contient un exemple en XML

# Reprendre les descriptions d'éléments dans un § de texte

Si vous écrivez (comme il est conseillé) un texte général (en prose) de présentation du modèle (à la manière des chapitres des *Guidelines*), vous pouvez y insérer les descriptions des éléments à l'aide de

`<specDesc/>` pouvant figurer dans une `<specList/>`

`<p>` Nous utilisons pour ce faire les éléments `<gi>app</gi>` et `<gi>rdg</gi>` :

```
<specList>  
  <specDesc key="app"/>  
  <specDesc key="rdg"/>  
</specList>  
</p>
```

# Références

*En complément des références fournies dans la présentation*

## Textes de référence

TEI Consortium, *TEI P5: Guidelines for Electronic Text Encoding and Interchange*,  
<<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html>>, particulièrement  
1 « The TEI Infrastructure »  
22 « Documentation Elements »  
23 « Using the TEI » (surtout, 23.3 « Personalization and Customization », 23.4  
« Conformance », 23.5 « Implementation of an ODD System »).

## Tutoriels

ODD : « **Getting Started with P5 ODDs** », *Guidelines...*  
Roma : « **Customizing the TEI with Roma** », *Guidelines...*

## Présentation et enjeux

Sebastian Rahtz & Lou Burnard, « Reviewing the TEI ODD system »,...

## Exemples

*Le meilleur moyen de comprendre ODD est encore d'en lire et d'en coder. Pour des exemples très riches, le mieux est probablement de télécharger les Guidelines, et de lire les fichiers ODD qui s'y trouvent (qui sont ceux du modèle TEI global). Voir : <https://github.com/TEIC/TEI>*