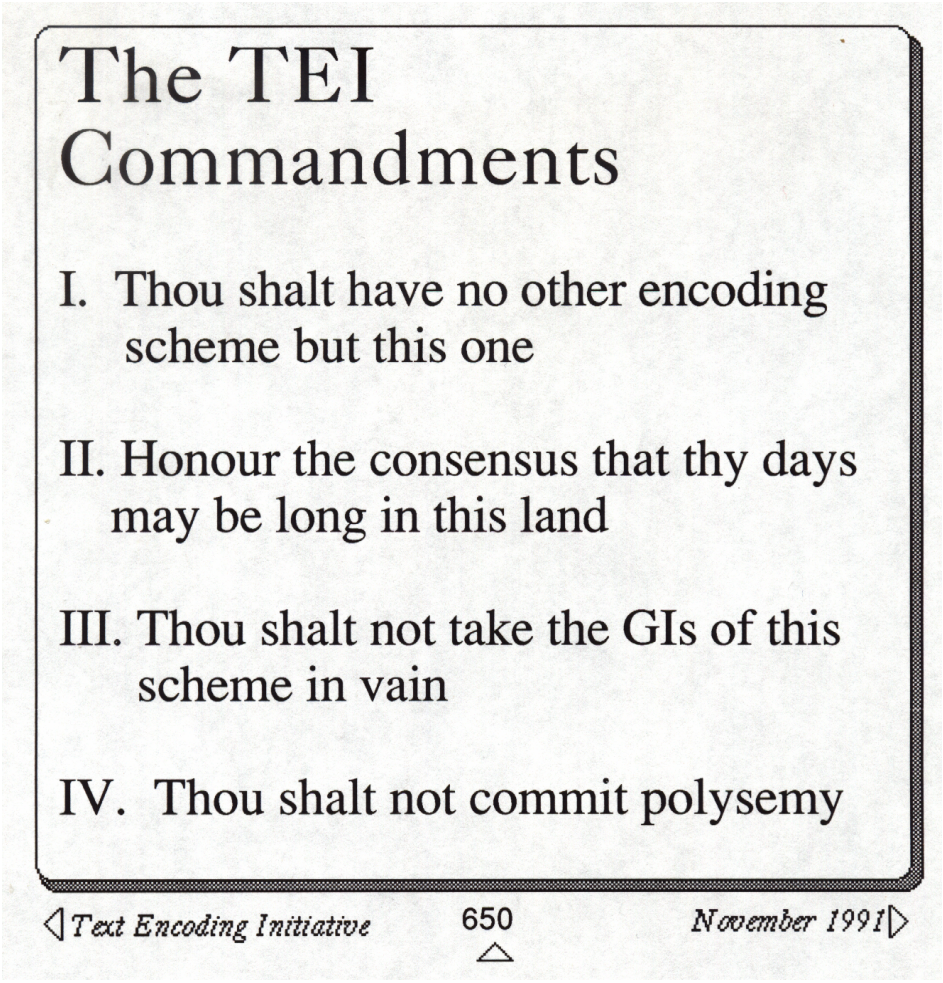
La personnalisation de la TEI

Un standard existe pour qu'on s'y conforme, non ?



L'esprit TEI

Qu'est-ce que cela veut dire : « être conforme » à la TEI ?

- une pratique de balisage consensuelle
- un lexique commun
- un respect de l'autonomie

La standardisation ne doit pas signifier « fais comme moi » ; elle veut dire « explique-moi ce que tu fais. »

… d'où les variations TEI

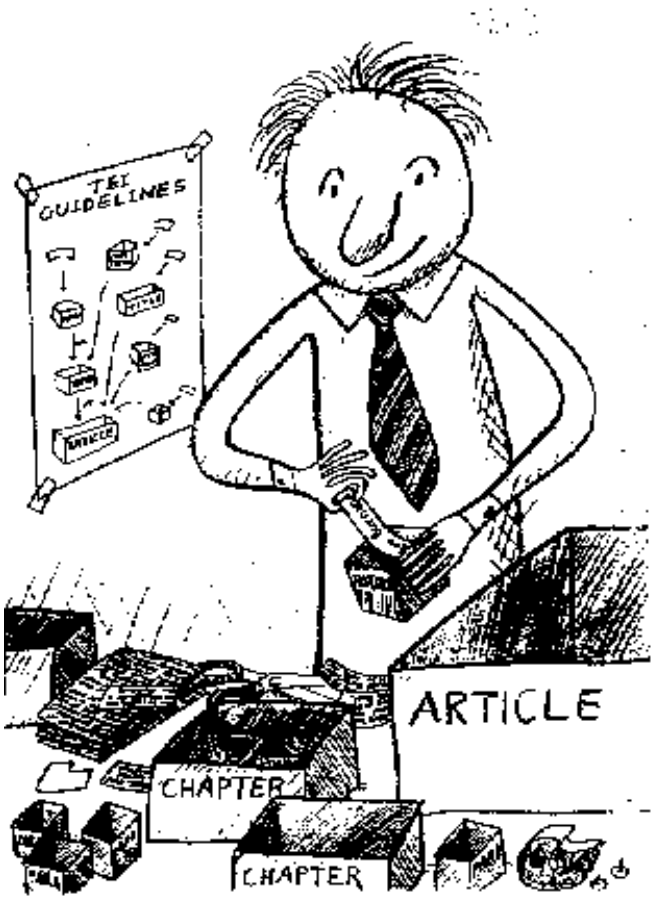
Par exemple : éléments pour description bibliographique : On a le choix entre

- <bibl> qui contient n'importe quel mélange de composants bibliographiques … ou aucun
- <biblStruct> qui contient une sélection prédéfinie d'éléments, strictement structurés
- <biblFull> qui est modélisé sur le <fileDesc> du TEI Header

```
<bibl xml:id="MK_73">Sturm, U. & Drang, F. :
<title>Musikalische Katastrophe</title>.
(Berlin, W. de Gruyter, 1973)</bibl>
```

```
<bibl  xml:id="MK73">
  <author  key="U. Sturm (1884-1919)">
    <persName>
      <surname>>Sturm</surname>
      <forename>Ulbricht</forename>
    </persName>
  </author>
  <author>Drang, F.</author>
  <title  xml:lang="de"  level="m">Musikalische Katastrophe </title>
  <pubPlace>Berlin</pubPlace>
  <publisher>W. de Gruyter</publisher>
  <date>1973</date>
</bibl>
```

La pérsonnalisation



Extrait du ICAME Journal, 1992

Le système TEI fournit un gabarit ou un kit lego pour la construction d'un système d'encodage bien adapté aux besoins de l'utilisateur, qui reste en même temps compréhensible par d'autres personnes ou systèmes, et dans lequel les modifications éventuelles se déclarent clairement.

Il n'y a pas de "TEI dtd"

- TEI est un système *modulaire*. On s'en sert pour créer un système d'encodage selon ses propres besoins, en sélectionnant des *modules* spécifiques
- Chaque module définit une brique contenant un groupe d'éléments (et leurs attributs)
- on peut séléctionner les éléments souhaités, et même en changer des propriétés
- on peut y mélanger des éléments nouveaux, ou bien natifs ou bien d'autres standards

Un encodage TEI représente une modélisation spécifique de l'univers textuel

Un système modulaire de modélisation

Tout projet souhaitant profiter de la TEI doit donc faire des choix pour modéliser son projet par moyen d'un *schéma* TEI ce qui n'est pas un science exacte ...

- la définition du schéma s'opère au moyen d'aller-retour avec les sources textuelles
- première modélisation à partir d'un échantillon représentatif
- ensuite, correction ou renforcement du contrôle lors du passage à l'échelle
- Le processus d'élaboration du modèle est souvent donc un processus incrémentiel de 'trial and error'

De quoi a-t-on besoin ?

- quelque chose qui documente notre modèle, nous permettant d'expliciter nos choix éditoriaux
- quelque chose qui permette de définir un sous-ensemble de la TEI
- quelque chose qui permette de générer un schéma pour contrôler la validité de nos documents : c-a-d
  - quelles balises sont disponibles
  - dans quels contextes
  - avec quels attributs
  - avec quelles valeurs
  - sous quelles conditions
- d'outils informatiques pour transformer et gérer nos données

One Document Does it all (ODD)

C'est précisément l'objet de ODD qui propose un vocabulaire spécialisé pour définir

- des schémas
- des types d'éléments XML
- des macros ou patrons
- des classes d'éléments ou d'attributs
- une manière de faire référence à ces objets

module tagdocs

–> Le language ODD est le vocabulaire XML utilisé par la TEI pour se décrire elle même.

Architecture globale de la TEI (rappel)

- La TEI est composé de 23 *modules*
- Chaque module déclare plusieurs *éléments* et quelques *classes*
  - une *classe d'attribut* définit un ensemble d'attributs
  - une *classe modele* regroupe des éléments ayant les memes propriétés structurals
- Un élément déclare les classes desquelles il est membre

ODD: les éléments de base

Un document ODD est un document XML-TEI comme tous les autres; comportant un en-tête TEI, un élément <body>, <back> etc, mais aussi enrichi d'un élément <schemaSpec> qui contient plusieurs déclarations.

- <schemaSpec> définit et identifie un schéma
- <moduleRef> fournit un ensemble de spécifications d'éléments en faisant référence à un module
- <elementSpec> fournit une définition d'élément (entière ou partielle)
- <elementRef> utilise une définition d'élément existant
- <classSpec> fournit la définition d'une classe
- <classRef> utilise une définition de classe existante

Sélection des modules : <moduleRef>

- L'attribut @*key* doit identifier un module TEI
- L'attribut @*source* indique la version du module, si nécessaire
- Par défaut, tous les éléments spécifiés par le module sont inclus à moins que...
  - l'attribut @*include* fournisse une liste des éléments spécifiques, ou
  - l'attribut @*except* fournisse une liste des éléments à exclure

Par exemple ...

Supposons un projet de transcription collaborative, où nous souhaitons utiliser

- une gamme très réduite des balises TEI
- une liste très contrainte de possibilités de valeur pour quelques attributs
- un élément pas encore prévu par la TEI

L'usage d'un schéma XML pour renforcer ces contraintes simplifie énormément la création d'une interface ergonomique, bien adaptée aux utilisateurs prévus

Exemple fictif (1)

```
<body>
  <head>Une personalisation TEI pour la
transcription collaborative</head>
  <p>Cette personalisation propose un
schéma minimal pour la transcription
collaborative
     des documents archivals. </p>
  <schemaSpec ident="transMin"
    start="TEI text div" docLang="fr">
    <moduleRef key="tei"/>
    <moduleRef key="header"
      include="teiHeader fileDesc
titleStmt publicationStmt sourceDesc"/>
    <moduleRef key="textstructure"
      include="TEI text body div"/>
    <elementRef key="ab"/>
    <elementRef key="pb"/>
    <elementRef key="unclear"/>
    <elementRef key="hi"/>
    <elementRef key="name"/>
    <elementRef key="title"/>
    <classRef key="att.global.rendition"
      except="rendition style"/>
    <classSpec type="atts"
      ident="att.declaring" mode="delete"/>
    <classSpec type="atts"
      ident="att.edition" mode="delete"/>
    <classSpec type="atts"
      ident="att.editLike" mode="delete"/>
  </schemaSpec>
</body>
```

- un peu de documentation
- un <schemaSpec> identifiable, précisant une langue de documentation et des éléments racine
- une séléction d'éléments
- suppression de plusieurs attributs

Exemple fictif (2)

```
<body>
  <head>Une personalisation TEI pour la
transcription collaborative</head>
  <p>Cette personalisation propose un
schéma minimal pour la transcription
collaborative
     des documents archivals. </p>
  <schemaSpec ident="transMin"
    start="TEI text div" docLang="fr">
<!-- ... -->
    <élémentSpec ident="hi"
      mode="change">
      <attList>
        <attDef ident="rend"
          mode="replace">
          <valList type="closed">
            <valItem ident="underline"/>
            <valItem ident="superscript"/>
          </valList>
        </attDef>
      </attList>
    </élémentSpec>
<!-- ... -->
  </schemaSpec>
</body>
```

- la spécification existante pour <hi> est modifiée
- la spécification de son attribut *@rend* est remplacée
- la liste des valeurs possibles pour cet attribut est fermée

Exemple fictif (3)

```
<body>
  <head>Une personalisation TEI pour la
transcription collaborative</head>
  <p>Cette personalisation propose un
schéma minimal pour la transcription
collaborative
     des documents archivals. </p>
  <schemaSpec  ident="transMin"
    start="TEI text div"  docLang="fr">
<!-- ... -->
    <élémentSpec  ident="botName"
      ns="http://monexcellentprojet.com">
      <desc>nom botanique</desc>
      <classes>
        <memberOf  key="model.phrase"/>
        <memberOf  key="att.global"/>
      </classes>
      <content>
        <macroRef  key="macro.paraContent"/>
      </content>
    </élémentSpec>
<!-- ... -->
  </schemaSpec>
</body>
```

- nous ajoutons une spécification pour un élément non-TEI, appartenant à une autre espace de nommage
- cette spécification comporte
  - une description
  - une indication des classes TEI auxquelles l'élément appartiendrait
  - une indication de son contenu possible

Référence à des modules

```
<schemaSpec ident="monSchema"
 source="http://www.tei-
c.org/release/xml/tei/odd/p5subset.xml">
 <moduleRef key="tei"/>
 <moduleRef key="core"/>
 <moduleRef key="header"/>
 <moduleRef key="textstructure"
  except="div1 div2 div3 div4 div5 div6 div7 group"/>
 <moduleRef key="namesdates"
  include="persName placeName"/>
</schemaSpec>
```

Contraindre la valeur d'un attribut

```
<attDef ident="agent" mode="change"
 usage="req">
 <datatype minOccurs="1"
  maxOccurs="unbounded">
  <dataRef key="teidata.enumerated"/>
 </datatype>
 <valList type="closed" mode="replace">
  <valItem ident="fire">
   <desc xml:lang="fr">endommagé par le feu</desc>
  </valItem>
  <valItem ident="moisture"/>
  <valItem ident="rubbing"/>
  <valItem ident="smoke"/>
  <valItem ident="tear"/>
  <valItem ident="water"/>
  <valItem ident="unknown"/>
 </valList>
</attDef>
```

Les différentes manières de personnaliser la TEI

1. rédiger une spécification de haut niveau
2. utiliser les sous-modules de la TEI et spécifier dans ces sous-ensembles les fonctionnalités à activer

Le processeur ODD

Le processeur ODD, sera un outil logiciel capable

- d'assembler les composants référencés de la TEI
- résoudre des déclarations multiples
- vérifier la validité
- émettre un schema dans un ou plusieurs langages formels
- produire un document XML avec la documentation des composants

oXygen peut le faire... mais il y a aussi Roma