INITIATION À L'ENCODAGE DES TEXTES PATRIMONIAUX - FORMATION XML/TEI

SUFCO/CESR

TOURS, 3-5 NOVEMBRE 2010

PROGRAMME DE LA FORMATION

Mercredi 3 novembre
9h-10h30: Introduction à l'XML/TEI (Lou Burnard)
10h45-12h: Structuration de document (Nicole Dufournaud)
14h-16h: Métadonnées 1 (Laëtitia Bontemps)
16h15-18h: Métadonnées 2 (Laëtitia Bontemps)

Jeudi 4 novembre
9h-10h30: Encodage des textes patrimoniaux: l'imprimé 1 (Nicole Dufournaud)
10h45-12h: Encodage des textes patrimoniaux: l'imprimé 2 (Laëtitia Bontemps)
14h-16h: Encodage des textes patrimoniaux: le manuscrit 1 (Lou Burnard)
16h15-18h: Encodage des textes patrimoniaux: le manuscrit 2 (Lou Burnard)

Vendredi 5 novembre
9h-10h30: Les projets TEI (Laëtitia Bontemps) et Outillage TEI 1 (Lou Burnard)
10h45-12h: Outillage TEI 2 (Lou Burnard)
13h-15h: Indexation (Nicole Dufournaud)
15h-17h: Exploitation, analyse, et visualisation (Nicole Dufournaud)
17h-17h30: Conclusion et bilan

---

3 novembre

9h-10h30

Lou Burnard: Initiation à l'XML/TEI

PLAN:
1- qu'est-ce que l'encodage textuel?
2- quels sont les concepts fondamentaux d l'xml/TEI?

Introduction

• aujourd'hui, nouvelles attentes:
-utiliser des Bdd et instruments de recherche (cf.projet Isidore)
-coupler ces instruments avec des textes
-intégrer ces instruments dans une sorte de mère-porteuse numérique
-donner l'accès aux données

=> pour toutes ces attentes, la TEI peut aider, parce que:
-c'est un modèle consensuel bien établi
-elle permet la conversion des données
-elle permet l'intégration de données déjà existantes mais éparpillées
-elle s'appuie sur des formats et des technologies ouverts
-elle bénéficie d'une ontologie textuelle explicite

• qu'est-ce qu'un texte?
- un texte n'est pas la même chose qu'un document
- l'essentiel d'un texte ne réside pas dans l'apparence des lettres ou dans la mise en page
- l'essentiel d'un texte n'est pas sa version originelle
- l'essentiel du texte, ce sont ses interprétations, ses lectures
=> le texte = quelque chose d'abstrait, qui est une construction, une production d'une communauté de lecteurs
encodage = explicitation de cette abstraction afin de mieux la gérer
la TEI donne des concepts, des idées abstraites, sous forme de ressources réelles

1- qu'est-ce que l'encodage?

- un texte est plus qu'une séquence de caractères encodés
- un texte est plus qu'une suite de formes lexicales
=> encodage = nécessaire
l'effet Babel: pour la plupart des textes, il y a plusieurs lectures, et donc plusieurs manières de les exprimer et plusieurs manières de les encoder
bonne nouvelle: il y a des logiciels qui peuvent traduire d'un format à l'autre. mais le mieux, c'est quand même de définir un standard commun d'échange
= le balisage

pour le balisage, c'est mieux de séparer la forme et le contenu => balisage réutilisable, adaptable...

l'encodage peut porter sur des points très différents:
- on peut encoder l'apparence du texte (par ex. les retours à la ligne) => balisage très descriptif
- on peut encoder la structuration du texte (par ex. «c'est une épître») => balisage plus analytique
- on peut encoder le texte du point de vue de la linguistique (par ex. en encodant les articles, les substantifs, etc.)
- on peut encoder le texte du point de vue de l'histoire de la langue (par ex. en donnant, pour chaque mot différent d'aujourd'hui, son équivalent contemporain)

le langage d'encodage sert à
- spécifier les caractéristiques d'un texte
- expliciter les structures d'un texte
- linéariser un texte
- donner des méta-informations sur un texte
=> il faut choisir, parmi toutes ces fonctions, lesquelles sont les plus importantes pour le projet

---

quelques rappels sur xml...

• xml = façon de représenter des données structurées sous la forme d'une chaîne de caractères
l'xml ressemble à l'html, mais l'xml est extensible, doit être bien formé, et peut être valide
l'xml est indépendant de l'application, de la plate-forme et du vendeur
=> l'xml rend donc le pouvoir au fournisseur de données et facilite l'intégration de ressources diverses et polyglottes

• **principes essentiels**:
- un document contient au moins un *élément*

- un élément possède une balise d'ouverture, une balise de contenu (éventuellement) et une balise de fermeture
- un élément peut porter un attribut, chaque attribut portant lui-même un nom et une valeur
- un document xml est obligatoirement *bien formé* (c'est-à-dire qu'il suit la syntaxe xml)
- un document xml peut être *valide* (c'est-à-dire qu'il suit les règles du schéma)

• **structure** d'un document xml:
- un document xml est une arborescence composée de noeuds
- un seul noeud racine contient tous les autres
- chaque noeud peut être soit une arborescence, soit un élément (avec éventuellement des attributs), soit encore une chaîne de caractères
- chaque élément a un nom ou une identification numérique
- chaque attribut a un nom ou une valeur

• pour un document **bien formé**, il faut:
- qu'une seule racine contienne le document entier
- que chaque arborescence soit proprement imbriquée
- que chaque balise ouvrante ait sa balise fermante (à l'exception des balises de type <seg/>: combinaison ouvrant-fermant sans contenu)
- que les attributs sont entre guillemets (quelle que soit leur apparence)
- qu'il n'y ait pas d'espace entre le chevron et le slash
- se souvenir que les noms sont sensibles à la casse

• pour que le document soit **valide**, il doit respecter le **schéma**
un schéma peut spécifier:
- le nom de l'élément racine
- le nom de tous les éléments légaux
- les noms et valeurs des attributs
- les règles concernant l'imbrication et le contenu des éléments
- etc...

attention: un schéma ne spécifie pas la signification sémantique des éléments
un schéma peut être exprimé en WSD (langage schéma du W3C), en RNG (iso Relax NG), ou en DTD (iso). La TEI se sert de RNG.

2- qu'est-ce que la TEI?

- *Text Encoding Initiative* ou *Text Encoding for Interchange*
- but: faciliter la création, l'échange et l'intégration de données textuelles sous un format informatisé (TEI a été développée alors que le www n'existait pas)
- la TEI s'adresse à la fois aux débutants (qui cherchent des solutions bien connues) et aux experts (qui cherchent de nouvelles solutions)
- ses principes: faire des recommandations; préférer des solutions générales permettant cependant des spécialisations
- la TEI aujourd'hui: les *guidelines*:

elles sont peu prescriptives
il y a consensus sur les distinctions significatives dans un vaste ensemble de matériaux textuels
2 volumes
P5; plusieurs langues possibles pour exprimer les schémas

- TEI c’est aussi, bien sûr, une communauté d’utilisateurs
- TEI = un système modulaire, adaptable selon les besoins: on peut inventer des éléments, en emprunter à d’autres standards, ou modifier les propriétés d’éléments déjà existants

l’envergure TEI:
- structuration basique de textes continus
- transcription diplomatique (manuscrits...)
- données formelles
- données para- et métatextuelles
- données linguistiques
- corpus de schémas

la TEI est 100% standard xml: il faut voir la TEI comme **une puce sur l’éléphant xml**

outils TEI à disposition: Oxgarage, Roma, Xaira...

TP de mise en jambes: manipulation d’Oxygen

10h45-12h

Nicole Dufournaud: Structuration de document

PLAN:

1- pourquoi structurer?

2- différences entre livre et manuscrit

3- principes de balisage

4- méthodologie

Introduction

trois formateurs, trois disciplines (linguistique, littérature, histoire), mais un seul et même outil pour tous.

=> la TEI offre donc bien des principes généraux pour une application la plus large possible

1- pourquoi structurer?

déf.:structurer des documents digitalisés = convertir des données textuelles dans un format exploitable par la machine

en fait, 3 questions: pourquoi structurer? pourquoi ne pas utiliser de Bdd? pourquoi encoder?

- pourquoi structurer? tout part de l'ecdotique: les données textuelles, c'est compliqué
- pourquoi ne pas utiliser de Bdd? parce que la Bdd, c'est:

- l'explosion de l'information en champs et en tables, l'atomisation du texte
- la reconstruction du texte par des requêtes
- l'absence de fichier contenant l'ensemble du texte structuré
- l'absence d'une édition incrémentale et cumulative
- un retour au texte impossible

=> la Bdd, c'est mal!

- pourquoi encoder? plusieurs réponses:

- pour «encoder pour encoder»: cf. Buffon et ses classements; et de fait, parfois on encode, et on trouve après-coup l'utilité d'avoir encodé; mais encoder pour encoder, cela permet aussi de structurer la pensée, cela apporte une aide intellectuelle
- pour éditer en ligne (ce qui pose la question «comment faire?»)
- pour analyser les textes (ce qui pose la question des outils nécessaires à l'exploration des données)

2- différences entre livre et manuscrit

la structure du manuscrit n'est pas celle du livre imprimé

en fonction du support, il faut rendre compte:

- des divisions (chapitre, paragraphe... =>)
- de la mise en valeur typographique
- de la structure physique (reliure, déchirures...)

=> tout commence donc par l'étude de la structure

puis, pour structurer, il faut baliser

3- principes de balisage

balisage formel et typographique:

 et <head>

<p>

<lb/> (comprendre *line beginning* plutôt que *line breack*)
<fw> (*formwork*: en-tête et pied de page)
balisage sémantique:
<persName>, <geoName>, <date>, <address>...
attribut:
paramètre associé à un élément
@type, *@n*, *@rend* (italique, *bold*, etc.)
attention: certains éléments sont aussi des attributs (par exemple <label> et @label) – mais là, Lou Burnard proteste et dit que ce n'est pas censé se produire, et qu'il faut remédier à cela...

4- méthodologie

```
<teiHeader>
....
</teiHeader>
<text>
    <front> .... </front>
    <body>
        <div>
        ....
        </div>
    </body>
</text>
```

et éventuellement: <back> .... </back>

TP 1 et 2: du texte au document
- TP 1: sonnet de Du Bellay
poser les balises , <head>, <lb/>, <lg>, <l>
poser les attributs *@rend = «it»* et *@rend = «center»*
- intermède sur la structure: le sonnet dans l'oeuvre de Du Bellay
- TP 2: lettre d'Elie Vinet
poser les balises , <opener> (ouverture de lettre), <ab> (= bloc de texte indistinct), <lb/> (et numéroter), <closer> (fin de lettre; a pour enfants <salute> et <signed>)
poser les attributs *@rend = «center»* et *@n = «x»*

Remarque: <!.....> marque un commentaire: cela permet d'éviter le rouge dans l'éditeur xml

14h-16h
Laëtitia Bontemps: Métadonnées (1)

déf.: métadonnée = donnée numérique qui sert à représenter et à décrire une autre donnée (numérique ou non); = informations sur le document
les métadonnées sont présentes dans l'en-tête du fichier numérique, ou dans un fichier indépendant
rôle d'indexation des métadonnées: si elles sont normalisées, il y a interopérabilité, possibilité d'échange de données

on met les métadonnées dans <teiHeader>
l'en-tête TEI présente une description structurée du contenu du fichier xml; il y a des balises minimales obligatoires
....Desc / ....Decl / ....Stmt

4 sous-ensembles de balises:
- <fileDesc>: description bibliographique du fichier
- <encodingDesc>: description du projet et des choix éditoriaux d'encodage
- <profileDesc>: description des aspects non bibliographiques (circonstances de production, langue, sujet...)
- <revisionDesc>: historique des révisions du fichier

• <revisionDesc> contient <change>, <name>, <date>...

• <profileDesc> contient <langUsage>, <language>, <textclass>; *@ident*; iso639 pour codifier les langues

• <encodingDesc> contient <projectDesc>, <editorialDesc> (qui lui-même contient <correction>, <normalization> et <hyphenation>), <tagsDecl> (pour l'analyse des balises), <rendition> (pour les principes de restitution>

•<fileDesc> contient <titleStmt>, <editionStmt>, <extent> (poids du fichier en octets), <publicationStmt>, <sourceDesc>, <serieStmt> (collection), <notesStmt>

- <titleStmt> contient <title>, <author> (auteur du fichier), <respStmt>
- <editionStmt> contient <edition> (date et n°)
- <publicationStmt> contient <authority>, <address>, <availability>
- <sourceDesc> contient <biblFull> (éléments bibliographiques peu structurés) ou <biblStruct> (éléments bibliographiques très structurés), <listBibl>, <msDesc> (description des manuscrits; peut contenir <surrogates>: reproductions de la source)

16h15-18h
Laëtitia Bontemps: Métadonnées (2)

on rencontre souvent des cas problématiques => questions sur la TEI list.
par ex.:
- comment encoder la source traduite, sa langue, son titre, son traducteur?
=> 2 possibilités: balise <editor> (éditeur scientifique) dans <editionStmt>, ou bien balise <respStmt> (cf. Hyperdonat)
- pour encoder le volume?
=> balise <biblScope>
- pour encoder qu'un ouvrage a un lien très fort avec un autre (par ex. la traduction de *l'Art poétique* d'Horace traduit par Peletier du Mans et *l'Art poétique* d'Horace)?
=> balise <relatidItem>; par exemple ici <relatedItem type = «source traduite» xml: lang = «lat»>
- ouvrage bilingue?
=> poser plusieurs valeurs de *@xml:lang*, en les séparant par un espace
- lieux de publication et d'impression différents?
=> *@type* différents
- commentateur? illustrateur?
=> <editor>; <respStmt>. note: cela permet de différencier les différents rôles d'une même personne (par exemple un journaliste qui est aussi poète)

on peut parfois passer d'un langage à l'autre; par exemple, il y a passage de la TEI à DublinCore pour le transfert des données depuis BVH dans Europeana
quelques normes d'encodage: ISO (dates langues, etc.), MARC (bibliothèques), DublinCore (échange de données via des entrepôts OAI), EAD (manuscrits)... et TEI!
DublinCore: seulement 15 balises; pour partager ses fichiers avec Europeana par exemple, il faut définir les correspondances entre TEI et DC: chacun fabrique ainsi son fichier DC en vue du partage.
attention: DC ne s'intéresse pas au contenu, mais seulement au catalogage (idem pour EAD).

4 novembre

9h-10h30

Nicole Dufournaud: Encodage des textes patrimoniaux: l'imprimé (1)

Récapitulatif:
- l'organisation de la TEI repose sur les schémas et les modules
- la TEI permet d'encoder différentes structures: roman, lettre, essai, théâtre, poésie...
- la TEI repose sur xml; la validation du xml repose sur le schéma; le schéma est une description du contenu valide pour ce qui est des éléments et des attributs; un attribut a une valeur

Organisation de la TEI:
- modules = listes de contenus valides
- ces modules ont une extension
- tout ça génère un schéma

Il y a 21 modules: Analysis, Certainty, Core, Corpus, Dictionaries.... (cf. liste en ligne de G. Poupeau)
4 modules nécessaires: TEI, Header, Core, Textstruct

TP3:
- poser , <head>, <pb/>
- poser <fw> avec @*place* (valeurs: top-right, top-left, bot-center, -bot-right), @*rend* (valeur: it), @*type* (valeurs: pageNum, sign, catch)
- poser <hi> avec @*rend* (valeurs larger, it, ro)
- baliser les lettrines: <c rend = «lettrine»>

Rappel: pour mettre deux valeurs sur un seul attribut, rajouter la deuxième valeur juste après la première (dans les guillemets) et mettre un espace entre les deux

structure du roman par défaut:

```
<text>
    <body>
        <div>
            <head> .... </head>
            <p>
                <lb/>
                    .
                    .
                    .
```

quelques modules spécifiques et certaines de leurs balises:
- analysis: <interp>; *@ana*
- figures: <figure>, <table>
- namesDates: <persName>
- Textcrit: <app>
- Transcr: <facsimile>

structure du module Verse:

```
<text>
        <body>
                <div>
                        <head> .... </head>
                        <lg>
                                <l>
                                        .
                                        .
                                        .
```

TP4:
poser , <head>, <lg>, <l>, <hi>, <fw>, <persName>; @*rend* (valeurs it ou ro)

module Drama (théâtre):
- <castList>: distribution
- <castGroup>: liste de personnages
- <castItem>
- <castRole>

+ module Core:
- <sp>: monologue; contient <speaker>
- <stage>: scénique

le <front> correspond à: page de titre, dédicace, privilège, préface/prologue, sommaire/tables, pièces liminaires

pour encoder la page de titre, il faut les balises TEI:
- <titlePage> (toute la page)
- <docTitle> (zone de titre)
- <titlePart> (un ou plusieurs)
- <docAuthor>
- <docEdition> (mention de l'édition)
- <docImprint>
- <pubPlace>
- <publisher> (éditeur)
- <docDate>
- <imprimatur> (privilège)

TP5:
poser: <front>, <titlePage>, <lb/>, <docTitle>, <titlePart>, <docImprint>, <pubPlace>, <publisher>, <docData>, <imprimatur>, @*rend*

Rque: l'encodage, c'est une façon de recréer aujourd'hui le travail des imprimeurs d'hier.

encoder un lien vers une image dans une page de titre:
<figure>
    <graphic url = «.....» height = «.....» rend = «.....»/>
</figure>

divisions autres que page de titre:

    <head>
    <p>
=> la typologie des divisions est à définir pour chaque projet

encoder un texte en colonnes:
<list>
    <head>
        <item>

tableau:
<table>
    <row> / <line>
on peut spécifier le nombre de colonnes en ajoutant à <row> l'attribut @*cols*
        <cell>

insertion de pointeurs: <ref target = «♯....»>
la valeur de l'attribut *@target* renvoie à l'attribut *@ xml: id* de la division concernée

génération automatique de sommaires: pour BVH, logiciel XTF (moteur de recherche intégré et génération de sommaires

14h-16h

Lou Burnard: Encodage des textes patrimoniaux: le manuscrit (1: transcription)

PLAN:
1- les balises pour l'encodage des transcriptions
2- lettres, abréviations, ratures, *choice*
3- le point sur l'édition génétique
4- facsimilé, lien image/transcription

Introduction

les caractéristiques d'un texte primaire à exprimer:
- manière dont le texte est inscrit
- orthographe
- rendu formel original
- ponctuation originale
- abréviations
- ajouts et suppressions
- erreurs et omissions
- illustrations, dessins, graphiques
- structure logique
- et d'autres choses selon projet

objectifs:
- mettre à disposition du plus grand nombre les sources primaires
- les rendre compréhensibles
- établir un texte définitif à partir de plusieurs témoins, compléter les transcriptions avec un *stemma* et des notes d'apparat critique

=> une transcription devrait suivre des règles éditoriales explicites, elles-mêmes fondées sur les recommandations admises dans la discipline scientifique concernée

1- les balises pour l'encodage des transcriptions

module Core: <abbr>, <add>, <choice>, <corr>, <del>, <expan>, <gap>, <sic>
module Transcr (sert à restituer apparat critique): <addSpan>, <om>, <damage>, <damageSpan>, <dalSpa,>, <ex>, <facsimile>, <fw>, <handNotes>, <handShift>, <restaure>, <space>, <subst>, <supplied>, <surface>, <zone>
module Textcrit: <app>, <lacunaEnd>, <lacuneStart>, <lem>, <listWhit>, <rdg>, <rdgGrp>, <variantEncoding>, <wit>, <witDetail>, <witEnd>, <witStart>, <witness>

structure logique et mise en page: tensions => problèmes de chevauchement
=> utiliser les éléments vides de type *milestone*: <gb/>, <pb/>, <cb/>, <lb/>; on peut les identifier par *@n* et par *@xml: id*
!!!: <expan>: pour abréviations, brévigraphes... mais on n'édite pas la forme du manuscrit, seulement la forme restituée

<fw>: pour ce qui ne fait pas partie du texte (titre courant etc.); *@type*, *@place*
!! sert seulement pour ce qui est déjà présent dans le manuscrit

2- lettres, abréviations, ratures, *choice*

abréviations:
- par suspension (e.g. pour *exempli gratia*)
- par contraction (M. pour Monsieur)
- par brévigraphes (=> symboles Unicode, mais pas forcément présents dans toutes les polices)

comment restituer l'abréviation?
- ne pas le faire
- restituer par <abbr> et <expan>
- restituer par <om> (suppression d'une ou plusieurs lettres) et par <ex>
- restituer les deux au choix avec <choice> => choix d'afficher les restitutions ou les formes originelles

<abbr> peut avoir les attributs *@type*, *@cert* et *@resp* (pour préciser par exemple que l'abréviation est un brévigraphe lu avec tel degré de certitude par untel)

corrections: <sic> = leçon jugée erronée mais conservée quand même; <corr>: version corrigée; là encore, il peut y avoir <choice>

normalisation: <reg>: regularisation; <orig>: forme originale; <choice> est possible.
Rque: on peut aussi normaliser par des attributs (dates, mesures...) => cela permet une recherche

<add>: ajout; <del>: suppression; <subst>: substitution (groupe les deux)
on peut ajouter *@place* pour les ajouts interlinéaires, marginaux, etc.

<unclear>: *@reason*, *@cert*

<damage>: un dommage matériel a affecté le texte, mais au moins une partie du texte est lue avec certitude. *@agent*, *@extent*

<supplied>: restitution de lettres omises; il faut faire la différence entre un texte présent mais abîmé, et un texte oublié

<handShift>: reprise de main ou début d'une nouvelle séance d'écriture. *@script*, *@medium*, *@new*, *@resp*
exemples: <handShift medium = «black-ink»>
ou bien <handShift new = ♯ h1»>: description de la main; le ♯ est un pointeur vers le header, mais on peut aussi avoir un url entre les guillemets, sans ♯. ici, on aurait dans le header:
<handNote xml: id = «h1»> Carefully written etc.

pour marquer l'annulation des suppressions: <restore>

3- le point sur l'édition génétique

l'édition génétique prend en compte le texte en tant qu'il est en train de s'écrire. balises pas encore disponibles dans TEI, mais bientôt!

<mod>, <modeSpan>, <metaMark> (signes laissés dans la marge etc.), <used>, <undo>, <redo>, <rewrite>, <transposeGrp>, <transpose>
@stage: association possible du phénomène à une étape identifiée dans l'évolution du document; renvoie à <creation> dans <profileDesc>, dans <teiHeader>
et aussi structuration parallèle du document vu comme un objet physique composé de surfaces, de zones, de lignes... tout ça sera bientôt utilisable

4- facsimilé, lien image/transcription

seulement des images, et parfois zones d'intérêt dans ces images
=> besoin de faire le lien entre ces images et les parties de la transcription

<pb facs = «page1.png»>: solution simplissime; mais plus la situation est compliquée, moins ça fonctionne
Rque: *@facs*: renvoie à une image (url ou pointeur); à placer sur la partie qui correspond à l'image: , ou s'il y a plusieurs pages, <pb/>

<facsimile xmlbase = «http.....»>
    <graphic url = «page1.png»/>
        .
        .
        .
</facsimile>
Rque: *@xmlbase* permet de donner seulement une url générale, dans laquelle se trouvent les graphic url.

<surface>: pour regrouper des images équivalentes mais de résolution différente

lien image/transcription:
<facsimile> <graphic xml: id.....> etc.
et <text> <pb facs = ♯ page1>

pour définir les zones: utiliser coordonnées de l'espace cartésien

=> tout ça permet de paralléliser du texte et de l'image même sur des portions très étroites

16h15-18h

Lou Burnard: Encodage des textes patrimoniaux: le manuscrit (2: description)

but: découvrir les balises spécifiques pour l'encodage des descriptions de manuscrits

le manuscrit est un objet unique, mais il existe peu de conventions de description largement partagées

l'élément <msDesc> est apte à plusieurs applications:
- bdd de record bibliographique (*finding aid*)
- texte discursif contenant plusieurs records (*catalogue raisonné*)
- assemblage de métadonnées au sein d'une édition numérique (*electronic edition*)
- instrument pour la codicologie quantitative

<msDesc> peut apparaître au même endroit qu'un <p>, ou dans le <teiHeader>, attaché à l'édition numérique d'un ou plusieurs manuscrit(s).
<msDesc> doit contenir <msIdentifier>, qui lui-même contient
- le lieu: <country>, <region>, <settlement>
- l'entrepôt: <institution>, <depository>
- l'identifiant: <collection>, <no>

!!! <msName> = le «petit nom» du manuscrit
<msDesc> peut facultativement contenir:
- <msContents>: à l'intérieur, soit <p>, soit <msItems>, et éventuellement, avant ça, un sommaire
- <history>: historique complet et provenance du manuscrit; idem: soit <p>, soit balises: <origin> (qui appartient à la classe datable, et a donc des attributs spécifiques: *@when* etc.), <acquisition>, <provenance>
- <additional>: disponibilité, informations administratives... <adminInfo>, <custHist>, <note>, <recordHist>, <availability>; <surrogates>: verssions alternatives (numérisation, notice bibliographique...)
- <physDesc>: cf. ci-dessous

<msItem>: localisé par <locus>: où se trouve le passage dans le manuscrit; <author>, <respStmt>, <title>, <rubric>, <incipit>, <explicit>, <colophon>, <finalRubric>, <quote>, <textLang>, <decoNote>, <bibl>, <listBibl>, <note>, ou un autre <msItem> imbriqué (utile quand il manque des morceaux d'un manuscrit - ou alors, on met cette information dans <note>).

description physique du manuscrit:
- <objectDesc>: support matériel
- <handDesc>: description d'écritures
- <musicNotation>, <decoDesc>, <additions> (=annotations)
- <bindingDesc>, <sealDesc>: reliure, sceaux
- <accMat>: matériaux associés, attachés

```
<objectDesc>
        <supportDesc>
                <condition>
                        <damage> blabla
```

<objectDesc> rattaché à <supportDesc> ou à <layoutDesc>

dans <layoutDesc>:

<layout column = «...» ruledlines = «...»>

<material>

<foliation>

Rque: on peut spécifier plusieurs *layout*, parce que toutes les pages ne sont pas identiques pour ce qui est des colonnes, des lignes, etc.

<handDesc>

<handNote>

<decoDesc>

<decoNote>

<additions>: notamment lettrine transformée en dessin

!!! dans le cas où des manuscrits distincts à l'origine ont été combinés pour faire un manuscrit composé, le <msDesc> peut contenir un <msPart>, qui a la même structure que le <msDesc>, et qui est en fait un <msDesc> à l'intérieur du <msDesc>.

TP à partir d'une notice bibliographique

5 novembre

9h-10h30

Laëtitia Bontemps: Les projets TEI

et

Lou Burnard: Outillage TEI (1)

Laëtitia Bontemps: Les projets TEI

- Deutschestextarchiv: téléchargement de fichiers xml, consultation du facsimilé en mode image, consultation possible en mode texte
- Artamène ou le Grand Cyrus: schéma TEI lite, affichage texte/image au choix (mais image et texte ne sont pas alignés), notes en pop-up, téléchargement pdf/html/xml/word/ibook, outil Philologic (recherche par mot)
- Lettres de V. van Gogh: plus de 500 lettres transcrites et traduites en anglais, plusieurs modes recherche (correspondant, lieu, etc.), double affichage au choix (facsimilé/traductions/notes/ travail sur le texte/transcription respectant les fins de ligne; deux fenêtres, et on choisit avec un onglet ce que l'on veut afficher), moteur de recherche (Lucene)
- le projet Newton: moteur de recherche (xtf), accès aux transcriptions diplomatiques et normalisées, et au facsimilé, pas d'alignement texte/image
- chroniques latines du Mont Saint Michel (*cf.* journées Mutec): pas de moteur de recherche

Remarque: Hyperdonat n'est pas référencé sur le site de TEI...

Lou Burnard: Outillage TEI (1): Roma

but = personnalisation des schémas TEI

Roma est une application web

le système TEI comprend plusieurs modules

chaque module contient plusieurs spécifications d'éléments

chaque spécification contient:

- un nom canonique <gi>
- une description de sa fonction
- une déclaration pour chacune des classes auxquelles il appartient
- une définition pour chacun de ses attributs
- une définition de son modèle de contenu
- des exemples d'usage, des notes, des liens...

une spécification de schéma TEI (<schemaSpec>) se fait par la sélection de modules et d'éléments, avec éventuellement des modifications

un document TEI contenant une spécification de schéma est un ODD (One document does it all)

qu'est-ce qu'un module? c'est une manière convenable de regrouper un ensemble de déclaration d'éléments (*cf.* plus haut: analysis, certainty, core...). On l'associe d'habitude avec des champs de travail spécifiques.

on crée un schéma TEI en faisant une sélection parmi ces modules.

comment choisir? plusieurs possibilités:

- on prend tout => pas pratique
- on part d'une sélection prédéfinie (TEI lite, TEI bare...)
- on peut procéder de façon artisanale (mais dans ce cas, il faut être conscient de toutes les possibilités)

=> Roma est un logiciel en ligne qui simplifie cette procédure

démonstration et TP Roma
onglet «modules»: pour voir la liste des modules présents dans <schemaSpec>; on peut ajouter/supprimer/modifier des modules
TP: contraindre les valeurs légales de *@type* dans ; ajouter un nouvel élément <saintName>; insister sur le fait que chaque contient un seul <lg>
(Remarque: il y a plusieurs langages pour exprimer un schéma: iso relax NG, W3C schéma langage, xml DTD langage
modèles de contenu: RelaxNG
datatypes: exprimé par W3C datatypes)
il est nécessaire d'utiliser les classes. La TEI contenant plus de 500 éléments, il est nécessaire d'utiliser des classes.
2 type de classes existent: attribut (chaque membre de la classe porte les mêmes attributs), et modèle (chaque membre peut apparaître dans la même localisation hiérarchique). Les classes sont elles-mêmes classables en classes. Un élément peut appartenir à plusieurs classes sans distinction de son module d'origine.

- attribute classes: elles portent un nom qui commence par «att»; att. naming contient *@key* et *@ref*; att. typed contient *@type* et *@subtype* => quand on crée un nouvel attribut, on n'a pas besoin de la redéfinir: il suffit de dire à quelle classe il appartient.
- model classes: model.pLike contient des membres qui ressemblent au <p>, qui se mettent au même endroit etc.; au contraire, les membres de model.pPart peuvent être contenus dans un <p>. model.pPart contient

* model.pPart.edit (<corr>, <del>)
* model.pPart.data (<date>, <name>)
* model.pPart.msDesc (<origPlace>)

DONC pour créer <saintName>, il faut se poser trois questions:
il ressemble à quoi? => <persName>
il peut contenir quoi? => seulement du texte
il peut apparaître où? => dans le titre, une phrase, un paragraphe, mais pas entre
=> c'est un membre de la classe model.nameLike!

le contenu d'un élément est défini par rapport aux classes d'éléments, et pas par rapport aux éléments directement. Il y a des modèles prédéfinis très utiles:
- macro.paraContent: le contenu des éléments qui ressemblent au paragraphe
- macro.phraseSeq
- macro.xText: le contenu des éléments qui contiennent des caractères non unicodes (éléments <g>)

il est possible de contraindre la valeur des éléments ou la valeur des attributs en se servant des datatype. Pour répondre à cette possibilité, il y a plusieurs macro: data.word, data.name, data.language, *etc*. Ou bien on peut créer ses propres contraintes avec le langage Schematron... mais encore faut-il le maîtriser!

TP: faire son propre schéma

10h45-12h
Lou Burnard: Outillage TEI (2)

on peut transformer n'importe quel document tei xml avec une feuille de style xslt.
css: autre standard qui ne permet pas de transformer le document, mais d'ajouter des propriétés à chaque noeud.
la feuille de style sert
pour passer de l'ODD au schéma,
et pour générer une version online des *guidelines etc*.

le point sur un nouvel outil: OxGarage
il se sert des mêmes feuilles de style pour opérer des transformations à double-sens: odt vers tei et l'inverse, docx vers tei et l'inverse, docbook vers tei, tei vers ePub, teiP4 vers tei P5 *etc*.
principal intérêt: il permet de transformer des documents en xml; mais c'est seulement pour avoir un premier aperçu de ces documents.

dans une feuille de style, il y a plusieurs paramètres; il est possible de contrôler ces paramètres:
- directement sous Oxygen
- sur la ligne de commande
- avec un *stylesheet* local

13h-15h
Nicole Dufournaud: Indexation

comment repérer et baliser les entités nommées?
ces entités sont de plusieurs types: personnes et lieux
problèmes: orthographe et variantes, changement des noms de lieux, identité sociale...

aux débuts de la TEI, il y avait un seul élément: <rs> (referencing string); type «person» ou autre
puis <name> (type «person» *etc*.)
mais aujourd'hui, c'est plus précis:
- <geogName> (peut être typé) <name> </name> </geogName>
- <geogFeat>: enfant de <geogName> pour tous les noms communs liés à la géographie
- <placeName>: lieu plus précis et plus peuplé; enfants: <settlement>, <region>

TP6
placer <geogName>, @*type*, <country>, <orgName>, <name>

les noms de personne: <persName> a pour enfants:
- <surName>: nom de famille/patronyme
- <foreName>: prénom
- <genName> (l'aîné...)
- <roleName>: titre et avant-nom
- <addName>: épithète
- <nameLink>: particules nobiliaires
- <placeName>

TP 7
placer toutes ces balises, et aussi <geogName>, <time> et <date>

Remarque: si plusieurs personnes partagent un même nom: *@who* ou *@xml: id*

index et bdd:
index de personnes avec régularisation: <index> <term> *@indexName*
et à la fin de ce que l'on veut indexer: <index index Name = «indexperson»> <term> forme régularisée </term> </index>
problème: c'est une façon de faire récente qui présente l'inconvénient de faire rajouter quelque chose (la forme régularisée) dans le texte.

bdd de noms et d'événements:
<listPerson>
    <person> et *@xml:id*
    <relation> *@name*, *@mutual = «#....@»* (... = un xml id)
<listEvent>>
    <event> *@type*
autre problème: ce système n'est pas prévu pour être placé dans du texte: s'il y a du texte entre les balises, ça ne va pas => à utiliser plutôt dans un fichier externe
Présentation d'exemples tirés du projet Marillac

15h-17h
Nicole Dufournaud: Exploitation, analyse, et visualisation

la visualisation n'est pas la même chose que l'affichage.
cf. les travaux de J. Bertin: pas sur l'informatique, mais quand même sur la visualisation
problème: difficile à utiliser pour des données peu ou pas structurées

la TEI ne permet pas de montrer graphiquement des analyses => il faut utiliser d'autres outils.

Millefeuille: Archives Nationales/INRIA/Ecole des Chartes/Paris I
but: structurer les almanachs royaux afin d'en tirer une meilleure connaissance de l'administration de la France
logiciel: Eclipse. index automatique sous forme d'arborescences; entrée = forme régularisée, et à l'intérieur: occurrences
attention: indexation dans tous les fichiers à la fois
de plus le fichier original n'est pas pollué, car les index se créent dans d'autres fichiers; les balises sont posées dans d'autres fichiers. => dans le fichier principal, on a seulement un pointeur.

infozoom: outil d'exploitation des données
geneaquilts: pour constituer des arbres généalogiques
timeline: pour une frise généalogique; *cf*. projet Marillac

la structuration de données textuelles représente la fin de la division du travail; elle permet au chercheur de devenir un acteur, elle l'aide à réfléchir et à analyser, et elle permet l'émergence de nouvelles idées.

atouts du xml: il est pérenne; il peut être lu par un humain; il permet l'indépendance entre le contenu et l'outil.

17h-17h30
Conclusion et bilan

Idée de commencer par encodage des <persName> et autres <geoName> plutôt que par le <teiHeader>: balises plus concrètes, et donc plus accessibles... mais en même temps, la progression dans ce sens serait moins logique.
Nous voulons un module n°2 pour prolonger les acquis de ce stage!!