Création d'un schéma propre et d'une documentation

Un «bon» document .xml doit à la fois être *bien formé* (c'est-à-dire qu'il doit respecter la grammaire de base du langage, pour tout ce qui a trait à l'imbrication des éléments par exemple, et *valide*, c'est-à-dire être conforme à un certain nombre de règles précises que l'on va indiquer au document en y associant ce que l'on appelle un *schéma de validation*. Un document bien formé n'est pas nécessairement valide, car s'il respecte la syntaxe .xml, il peut outrepasser les règles qui sont dans le schéma (utiliser des balises qui ne sont pas incluses dans le schéma, ou utiliser un attribut dans un élément qui ne doit pas y figurer, etc).

J'aurais voulu apprendre au cours de cette année à créer un schéma de validation de mon document .xml sans l'aide de ROMA, l'outil mis en place par la TEI (je voudrais notamment pouvoir proposer une aide à l'encodage et la vérification automatique de la collation de tous les témoins dans chaque app, mais cela suppose de jongler avec les arrêts et lacunes de manuscrits : je ne sais pas si ce point est possible) ; cependant le manque de temps ne m'a pas permis de me pencher sur la question. J'ai donc utilisé cet outil qui est assez pratique pour normaliser les choix d'édition que j'ai pris au début de mon travail et m'y tenir. J'ai notamment fermé la liste des valeurs de l'attribut `@type` qui est rendu obligatoire, l'attribut `@wit` dans les éléments `<lem>` et `<rdg>` ; l'attribut `@type` dans les éléments `<note>` est de même rendu obligatoire. Pour éviter de perdre de l'information au niveau de la bibliographie et bien distinguer les références internes du texte (les sources de Castrojeriz) et les références externes (la bibliographie), je rends obligatoire l'utilisation de l'attribut `@type` dans les éléments `<ref>`.

La transformation en .pdf

Un grand intérêt de l'édition électronique est, bien que cela ne soit pas l'essentiel, de pouvoir automatiser la mise en page et surtout de l'uniformiser. Le .xml est un langage de *description* et non un langage de *mise en page*. Ainsi, il est possible de décrire un aspect du texte (un ajout par exemple) sans se soucier de la manière dont il va être rendu dans l'édition : en ce sens, le .xml permet en quelque sorte de proposer l'essence du texte sans se préoccuper a priori de la forme ; de la même manière il est très facile de modifier la mise en forme d'un aspect du texte marqué par un balisage en particulier (si l'encodage est bien réalisé, c'est-à-dire uniforme). Il s'agit, à mon avis, du grand avantage qu'offre un système qui sépare la description du document de sa présentation, comme le fait la TEI.

Les transformations XSL

Première feuille : transformation vers LaTeX Mon édition est une édition traditionnelle «papier» : il me faut donc passer du format .xml à un format qui puisse créer un pdf, c'est-à-dire le format LaTeX. Ce passage d'un format à un autre est rendu possible par un «sub-langage» .xml, le XSL, comme je l'ai dit plus haut. L'idée est donc ici de passer d'un langage de balisage *descriptif* à un autre langage de balisage *de mise en forme*.

De façon schématique, si on encadre un certain *texte* par une balise <exemple>, il s'agit de passer du langage de balisage .xml :

<exemple>*texte*</exemple>

à une instruction de mise en page LaTeX, pour le texte compris entre toutes les balises *exemple* :

\instruction{*texte*}

L'idée du xsl est qu'il permet d'aller chercher dans le document .xml – qui fonctionne, je le rappelle, à la manière d'un arbre, avec un élément racine et des noeuds de différents niveaux– un nœud ou un type de nœud [360] particulier sur lequel il va opérer une transformation. L'essentiel se joue à l'aide de la balise XSL<xsl : text> qui permet d'insérer du texte brut avant et après l'élément à transformer [361]. Sur l'exemple générique précédent, une règle de transformation (*template*) pourrait donner :

```
1 <xsl:template match="exemple">
2    <xsl:text>\instruction{</xsl:text>
3       <xsl:value-of select="."/>
4    <xsl:text>}</xsl:text>
5 </xsl:template>
```

La traduction de cette règle minimale donne :

- Les lignes 1 et 5, comportent la balise ouvrante et fermante de la règle. La balise ouvrante indique qu'il s'agit d'une règle (*xsl :template*) et donne le chemin d'accès vers l'élément à transformer («*match*», «trouve»). Ce chemin d'accès (*match="element"*) est ici simplifié.
- Les lignes 2 et 4 permettent de mettre en place l'instruction LaTeX : la balise <xsl :text> affiche le texte qu'elle entoure.
- La ligne 3 dit «affiche-moi la valeur de l'élément pointé en ligne 1 (le point «.» indique que l'on reste sur le même noeud que l'ordre d'accès match="exemple")».

En conclusion, la règle dit : «*affiche moi la valeur de tout élément* "exemple" *entre le texte* "\instruction{" *et le texte* "}"».

La règle ci-dessus schématise à grands traits le fonctionnement de ma feuille XSL ; celle-ci est cependant plus compliquée et permet notamment de discriminer les apparats, en différenciant les variantes superflues et les variantes qu'il faut mettre en valeur, ou bien de proposer divers niveaux de notes (deux niveaux de notes de bas de page distincts en fonction du type de la note, si elle commente la matérialité du texte ou son contenu, plus des notes marginales pour l'indication des fins de texte et lacunes – voir plus bas). J'ai dû donc par exemple recours à des règles conditionnelles.

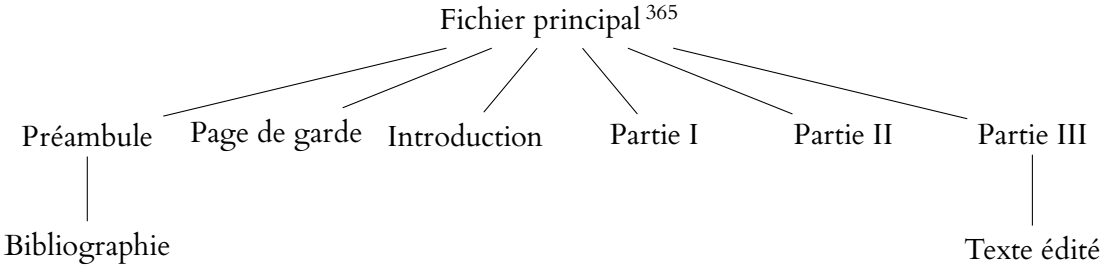
360. Il est difficile de trouver une définition du nœud : le wiki de l'Université de Genève nous dit «Un nœud est un composant particulier d'un document .xml». `http://edutechwiki.unige.ch/fr/N\T1\oeuds_:_xml_xhtml_et_DOM`.

361. Je remercie Ariane Pinche qui a commencé par m'indiquer ce point indispensable et qui m'a bien aidé.

Seconde feuille : espaces et caractères spéciaux La gestion des espaces est un des problèmes majeurs du .xml : les sauts de lignes dans un fichier .xml créent nécessairement des espaces non voulus. Pour pallier ce qui est un obstacle à la bonne présentation d'un texte, j'ai créé une seconde feuille de transformation qui à partir d'un document .xml très simple (composé d'une seule balise et du texte transformé et compilable par mon éditeur LaTeX), applique plusieurs transformations de caractères(«e» en «&» par exemple) et supprime les espaces non souhaités. Cette feuille m'a permis empiriquement de régler l'ensemble de ces problèmes de façon systématique et automatique. Je propose ces deux feuilles de transformation sur *github.com* [362].

Architecture et paquets principaux de mon édition LaTeX

Selon le type de travail, il peut être plus ou moins intéressant de proposer une feuille de transformation .xml vers LaTeX directement compilable ou non [363]. Ce n'est pas la méthode que j'ai choisie, et ce pour plusieurs raisons. Mon mémoire n'est pas une édition brute et comporte tout un ensemble de textes divers : il serait très très peu pratique de travailler directement sur la feuille de transformation. Il m'a paru donc nécessaire de créer un fichier indépendant comprenant mon édition, et de créer un lien de mon fichier LaTeX principal vers celui-ci. J'ai de même décidé de fragmenter mon travail en une myriade de sous-documents, ce qui me semble plus pratique pour travailler. Le fichier .pdf ci-présent est ainsi le résultat de la compilation de cet ensemble de fichiers LaTeX. J'ai choisi de faire de chaque partie de mon mémoire un fichier indépendant. Comme dit plus haut, mon fichier .pdf n'est pas produit à partir d'un seul document .tex, mais à partir d'un ensemble de sous-fichiers rattachés à mon document principal LaTeX [364]. L'architecture de cet ensemble est la suivante :

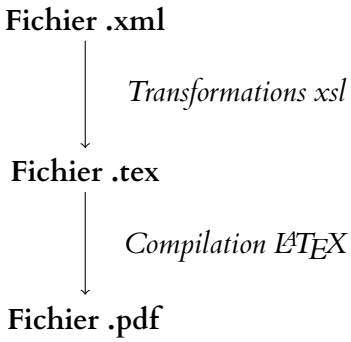


Les paquets principaux et indispensables de mon document LaTeXsont les suivants. Le package que j'ai utilisé pour gérer l'apparat de mon édition est nommé *ednotes* [366]. J'ai choisi *biblatex* pour ma bibliographie [367].

362. `https://github.com/mattlev2/CastrojerizB.git`.
363. Le travail de Marjorie Burghart, sur lequel je me suis beaucoup appuyé, va dans ce sens : la transformation xsl qu'elle propose produit un fichier LaTeX fonctionnel.
364. J'intègre en partant du fichier-mère (le squelette du texte) tous les autres fichiers, à l'aide de la commande `\input{fichier}` . Seule la bibliographie est insérée à l'aide du package `subfiles`
365. Comprend le texte de la page de garde, les remerciements, l'avant-propos et la conclusion.
366. `https://www.ctan.org/pkg/ednotes`.
367. `https://www.ctan.org/pkg/biblatex`.

Processus éditorial complet

Je propose par le schéma ci-dessous le processus éditorial complet, entre mon fichier .xml encodé selon les normes de la TEI P5 et le fichier .pdf final :



Pour l'édition proprement dite, cela a l'avantage de permettre de passer en quelques secondes de l'encodage .xml à un résultat visible.

Difficultés rencontrées

Cette section essaie de retracer les difficultés que j'ai pu rencontrer au cours de mon travail ; ces difficultés et obstacles sont de toutes sortes. Je sépare les problèmes d'édition et les problèmes informatiques pour clarifier au mieux ma présentation, tout en sachant que cette distinction est bien souvent inadéquate.

Le premier problème auquel j'ai pu être confronté est celui de la méthode d'établissement de l'édition : j'ai dû tâtonner tout au long de mon travail, entre édition progressive de tout le texte en comparant peu à peu chaque manuscrit, et une méthode que je trouve un peu plus pratique, où il s'agit d'encoder un bloc de texte avec en premier lieu le *codex optimus*, puis de faire la collation ensuite avec les autres textes. Dans tous les cas, le travail est extrêmement long et parfois très fastidieux.

La méthode qui a le mieux marché est la suivante : premièrement, établir le texte du *codex optimus* ; puis le comparer individuellement avec chaque témoin sur un segment de quelques propositions ; si une différence est trouvée sur le témoin que je compare, vérifier ce que lisent tous les autres témoins.

Problèmes d'ordre philologique

Problèmes de transcription : j'ai pu commettre un certain nombre d'erreurs au cours de ma transcription, erreurs dues à la taille de mon texte et au grand nombre de témoins que j'ai dû lire. Les erreurs les plus flagrantes et qui sont relativement faciles à corriger (du fait de la relecture constante du texte que requiert la collation) sont les sauts du même au même : il m'est arrivé de me rendre compte que j'avais fait le même saut du même au même qu'un témoin !