1 XML ALTO

1.1 Qu'est-ce qu'est le format ALTO ?

Le format XML ALTO s'est évolué à partir du projet européen METAe en 2003. Le projet qui a conditionné la création d'ALTO s'occupait du développement des logiciels dont des institutions patrimoniales pourraient servir à la création et à l'exploitation des fac-similés numériques de leur fonds. Le but du projet était d'extraire à partir des pages numérisées les informations portant sur la mise en page et les autres données structurelles. D'ici 2003, les logiciels OCR étaient déjà bien mis en pratique. L'enjeu à l'époque était d'élaborer un schème de données qui soumettrait le texte extrait à la logique structurelle de la page et du document. Tandis qu'un logiciel OCR reconnaître le texte du titre d'un chapitre et le texte de son sous-titre, un nouveau format devrait pouvoir distinguer les deux lignes de texte que le logiciel a reconnues et les hiérarchiser, en disant que le sous-titre est subordonné au titre du chapitre.

Le projet METAe a donc développé un format METS (*Metadata Encoding and Transmission Standard*) qui avait pour but d'augmenter les données textuelles extraites par un logiciel OCR avec la logique de la mise en page et du document. Bien que les logiciels OCR aient souvent exporté leurs prédictions dans un format du texte brut, le format METS visait à hiérarchiser le mélange des diverses données dans un format XML. Par exemple, à travers d'un arbre XML, le texte d'un sous-titre descendrait de la région dans laquelle les caractères de cette ligne de texte s'encadrent sur la page numérisée. De plus, au moins l'un d'eux, soit la donnée sur le texte du sous-titre, soit les données sur l'emplacement, porterait quelque chose pour dire que dans la logique du document il est un sous-titre.

Le schème METS a réussi à combiner les métadonnées de la ressource numérique ainsi que de l'objet text qui a été transcrit avec les données structurelles et textuelles de l'objet grâce au format hiérarchisé de l'XML. Mais le format primordial METS n'avait pas répondu à la question de comment bien structurer les dernières données, celles qui sont produites par un logiciel OCR ou HTR. Les créateurs du format ALTO ont décrit son prédécesseur comme un « emballage » (*wrapper*) pour la structure de données ALTO. [1] Tandis que le format METS organise les métadonnées et la logique du document, telles que l'occurence et l'ordre des pages, le format ALTO s'insère sous l'arbre de chaque page afin de décrire la transcription produite pour l'image numérique.

Normalement, un fichier XML ALTO décrit une page (ou une image) d'un document. Mais, comme se voit dans l'exemple donné dans l'exposition du schème quand il était nouveau en 2003, modélisé dans la Figure 1, l'élément `<Layout>` peut en fait contenir plusieurs éléments `<Page>`. [2] Néanmoins, la plu-

1. Birgit Stehno, Alexander Egger et Gregor Retti. « METAe–Automated Encoding of Digitized Texts ». In : *Literary and Linguistic Computing* 18.1 (1er avr. 2003), p. 77-88. ISSN : 0268-1145, 1477-4615. DOI : 10.1093/llc/18.1.77. URL : https://academic.oup.com/dsh/article-lookup/doi/10.1093/llc/18.1.77 (visité le 24/08/2022), p. 81.

2. Ibid.

part de logiciels HTR qui utilisent le format ALTO crée un fichier per page numérisée.

```
<Layout>
  <Page ID="XXX" PHYSICAL_IMG_NR="000" HEIGHT="000" WIDTH="000"
    STYLEREFS="XXX">
    <PrintSpace ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"
    >
      <TextBlock ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000
    ">
        <TextLine ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="
    000">
          <String ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="
    000" CONTENT="XXX"/>
          <Sp ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"/>
          <String ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="
    000" CONTENT="XXX"/>
        </TextLine>
        <TextLine ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="
    000">
          <String ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="
    000" CONTENT="XXX"/>
          <Sp ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"/>
          <String ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="
    000" CONTENT="XXX"/>
        </TextLine>
      </TextBlock>
    </PrintSpace>
  </Page>
  <Page ID="XXX" PHYSICAL_IMG_NR="000" HEIGHT="000" WIDTH="000"
    STYLEREFS="XXX">
    <PrintSpace ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000"
    >
<!-- ... -->
</Layout>
```

FIGURE 1 – La structure ALTO version 1, circa 2003

Le format ALTO est dans sa quatrième version, mais la structure actuelle ressemble bien au modèle qu'ont présenté les auteurs Birgit Stephno, Alexader Egger, et Gregor Retti en 2003. Dans sa première version, montrée dans la Figure 1, les éléments les plus petits étaient les segments de texte (`<String>`) et les espaces entre mots (`<Sp>`), balisés dans une ligne de texte (`<TextLine>`) qui appartient à un bloque de texte (`<TextBlock>`). Tous ces éléments XML porte un identifiant unique (`@ID`) et quatre coordonnées portant sur le rectangle dans lequel s'encadre le contenu de l'élément. Le contenu textuel est représenté dans l'attribut `@CONTENT` de l'élément `<String>`.

1.2 La structure actuelle des fichiers XML ALTO

Aujourd'hui, l'élément le plus petit d'une structure de données ALTO est un glyphe (`<Glyph>`), au lieu d'un segment de caractères (`<String>`). Par conséquent, dans le nouveau format, le contenu textuel est représenté deux fois, une

fois comme l'attribut `@CONTENT` de l'élément classique `<String>` et une deuxième comme le même attribut de l'élément `<Glyph>`. Un comparaison entre les deux arborescences est visualisé dans la Figure 3. Comme montre la sous-figure 2b, la nouvelle architecture se permet d'aller en plus de détail. Certains logiciels, tel que l'interface *eScriptorium*, produisent toujours les fichiers ALTO avec une variation de l'ancienne structure où l'élément `<String>` n'est pas répétable et représente le contenu textuel de la ligne.

En général, toute donnée portant sur la mise en place de la page se dispose de quatre coordonnées qui ensemble tracent un rectangle. Les valeurs des attributs `@HPOS` et `@VPOS` font les coordonnées x,y du point le plus haut à gauche du rectangle, comme se voit dans la Figure 2. La valeur de l'attribut `@HEIGHT` compte la différence entre la coordonnée y du point le plus haut et la coordonnée y du point le plus bas. La valeur de l'attribut `@WIDTH` calcule aussi la différence entre le côté gauche du carré et son côté droit. Ces quatre attributs sont attribués aux éléments `<PrintSpace>`, `<TextBlock>`, `<TextLine>`, `<String>`, `<Sp>`, et `<Glyph>`.

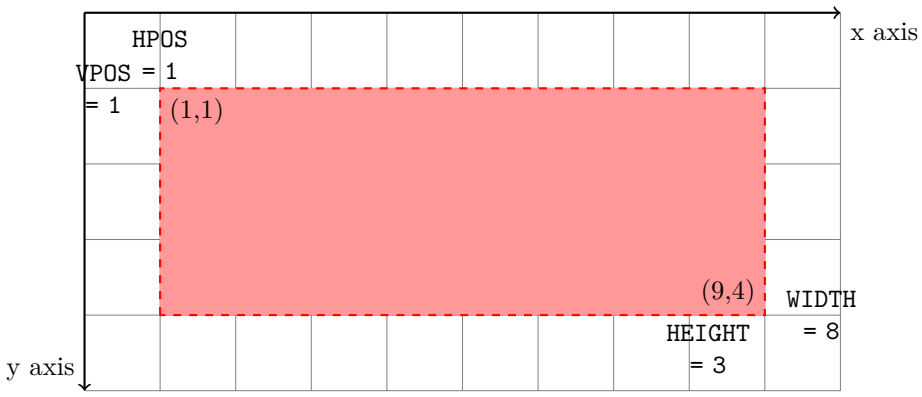
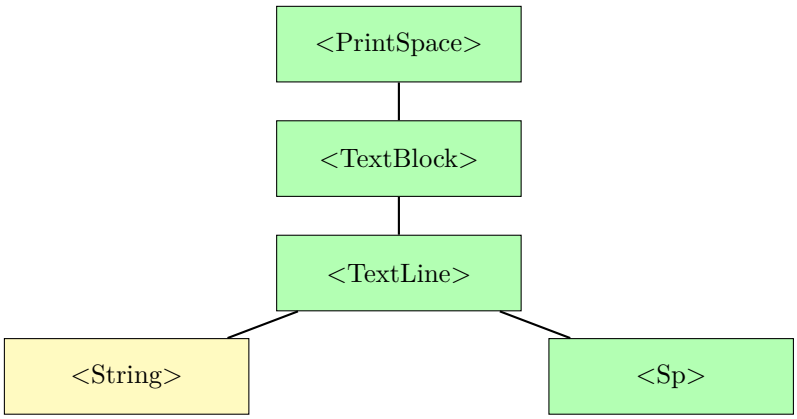
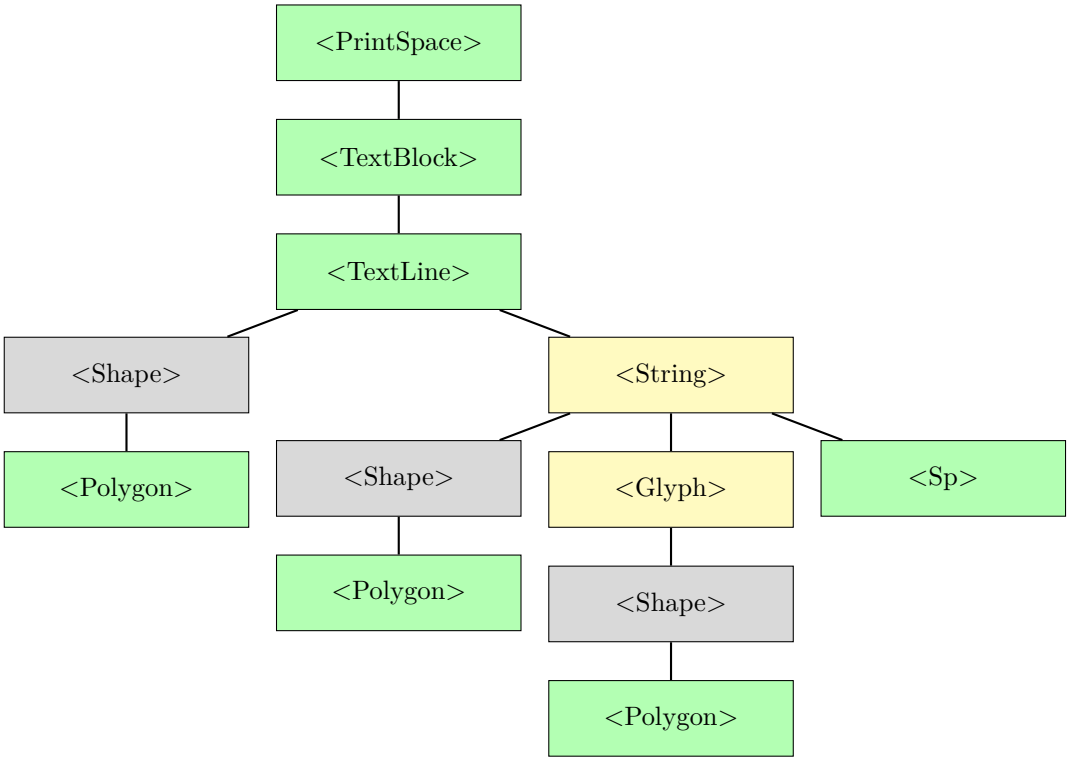


FIGURE 2 – Les coordonnées d'un masque en rectangle

L'arborescence actuelle du format ALTO diffère de l'original car, aujourd'hui, elle peut préciser les coordonnées d'un polygone en plus d'un rectangle. Cela est un développement dans la technologie des logiciels OCR et HTR. La visualisation dans la Figure 2b montre le nouvel élément `<Polygon>` qui descend directement d'un élément (`<Shape>`) qui lui-même ne porte pas d'attribut ni d'intérêt dans l'arborescence que de baliser les informations du polygone. Le contenu textuel est toujours balisé dans l'élément `<String>`, qui porte sur un segment ou mot d'une ligne de texte (`<TextLine>`). Mais en allant jusqu'au détail du glyphe dans l'arborescence actuelle, l'élément `<Glyph>` représente tout caractère composant un mot (`<String>`).



(a) Ancienne structure



(b) Nouvelle structure

FIGURE 3 – Modélisation des formats ALTO

Certains attributs actuels dans l'arborescence, l'attribut `@BASELINE` de l'élément `<TextLine>` et l'attribut `@POINTS` de tout élément `<Polygon>`, prennent comme valeur une chaîne d'entiers. Montrée dans la Figure 4, cette chaîne veut

représenter des pairs de coordonnées x,y. Chaque point articule une extrémité soit d'une ligne qui trace le baseline de la ligne de texte (`@BASELINE`) soit une ligne qui encadre la région reconnue par un modèle de segmentation (`@POINTS`). Le format ALTO encode chaque entier dans une chaîne dont les composants sont séparés par espace, où la valeur de l'axe des x précède la valeur de l'axe des y. Un polygone (`@POINTS`) peut avoir plusieurs points tout le long de son périmètre. Par contre, le baseline d'une ligne de texte (`@BASELINE`) compte toujours quatre entiers puisqu'il n'a qu'un début et une fin.

```
<Layout>
  <Page ID="XXX" PHYSICAL_IMG_NR="000" WIDTH="000" HEIGHT="000">
    <PrintSpace HPOS="000" VPOS="000" HEIGHT="000" WIDTH="000">
      <TextBlock ID="XXX">
        <TextLine ID="XXX" HPOS="000" VPOS="000" HEIGHT="000" WIDTH="
    000" BASELINE="1 2 3 4">
          <Shape>
            <Polygon POINTS="1 2 3 4 5 6 7 8"/>
          </Shape>
          <String ID="XXX" CONTENT="AB" HPOS="000" VPOS="000" WIDTH="
    000" HEIGHT="000" WC="1.0">
            <Shape>
              <Polygon POINTS="1 2 3 4 5 6 7 8 9 10"/>
            </Shape>
            <Glyph ID="XXX" CONTENT="A" HPOS="000" VPOS="000" WIDTH="
    000" HEIGHT="000" GC="1.0">
              <Shape>
                <Polygon POINTS="1 2 3 4 5 6 7 8"/>
              </Shape>
            </Glyph>
            <Glyph ID="XXX" CONTENT="B" HPOS="000" VPOS="000" WIDTH="
    000" HEIGHT="000" GC="1.0">
              <Shape>
                <Polygon POINTS="1 2 3 4 5 6 7 8"/>
              </Shape>
            </Glyph>
          </String>
        </TextLine>
<!-- ... -->
      </TextBlock>
    </PrintSpace>
  </Page>
</Layout>
```

FIGURE 4 – La structure ALTO version 4, circa 2022

2 XMl-TEI

La raison pour laquelle la TEI a été choisi pour fusionner toute donnée du pipeline *Gallic(orpor)a* est parce qu'il est un format XML souple, qui peut s'adapter facilement à plusieurs types de document et d'édition numérique. Cela veut dire qu'une exposition détaillée du schème TEI n'est pas possible. La ma-

nière pour encoder une ligne de texte n'est pas aussi fixée que cela du schème ALTO, par exemple.

Tandis qu'une ligne de texte dans un fichier ALTO est balisée dans l'élément `<TextLine>`, une ligne de texte dans un document TEI peut être encodé dans plusieurs façons. Elle peut suivre l'élément vide `<lb/>` ou elle peut être à côté d'autres lignes de texte, toutes balisées ensemble dans un élément tel que `<p>` ou `<div>`. De plus, parce que la TEI permet de classer les composants d'une ligne de texte selon la logique du document ou de la langue, certains mots ou phrases peuvent être balisés dans d'autres éléments, tel que l'élément `<date>` pour une encoder une année. Dans le même ordre des idées, une ligne de texte peut aussi être balisée dans les éléments qui expliquent sa fonction dans le document. Par exemple, une ligne de texte peut être un item dans une liste (`<item>`) ou la salutation à la fin d'une lettre (`<salute>`).

12 août 2022

Coucou ! J'ai fait une réservation pour ton anniversaire.

À demain ma chérie,

(a) L'exemple d'une lettre

```
<TextLine ID="line1" HPOS="0" VPOS="0" HEIGHT="40" WIDTH="200" BASELINE
    ="0 40 200 40">
  <Shape>
    <Polygon POINTS="...."/>
  </Shape>
  <String ID="seg1" CONTENT="À" HPOS="..." VPOS="..." WIDTH="..."
    HEIGHT="..." WC="1.0">
  <!-- ... -->
  <String ID="seg2" CONTENT="demain" HPOS="..." VPOS="..." WIDTH="..."
    HEIGHT="..." WC="0.888">
  <!-- ... -->
  <String ID="seg3" CONTENT="ma" HPOS="..." VPOS="..." WIDTH="..."
    HEIGHT="..." WC="0.9">
  <!-- ... -->
  <String ID="seg3" CONTENT="chérie," HPOS="..." VPOS="..." WIDTH="..."
     HEIGHT="..." WC="0.91">
  <!-- ... -->
</TextLine>
```

(b) La dernière ligne de la lettre encodée dans le schème ALTO

```
<salute>À <date when="2022-08-13">demain</date> ma <name ref="#Kelly"
    type="person">chérie</name>,</salute>
```

(c) La dernière ligne de la lettre encodée dans le schème TEI

FIGURE 5 – Le comparaison de l'encodage d'une ligne de texte en ALTO et TEI

Prenant l'exemple d'une lettre, la Figure 5 montre l'encodage de sa salutation dans les deux schèmes, ALTO et TEI. On voit que l'ALTO excelle à préciser l'emplacement des mots (et des caractères) sur la page d'un document. Mais après la reconnaissance du lettre, dont la certitude du modèle se représente par l'attribut `@WC`, le schème ALTO ne donne pas d'autre information. L'encodage dans le format TEI, par contre, enrichit la ligne de texte avec beaucoup d'information. Grâce à l'élément TEI `<salute>` on sait que la ligne de texte est la salutation d'une lettre ou quelque autre forme de communication. De plus, l'encodage appuie sur la date en-tête pour attribuer au mot *demain* une date précise qui est encodée dans l'attribut `@when`. En fin, le schème TEI dispose d'un système pour réunir les occurrences du même concept dans un texte, tel qu'une personne. L'encodage dans la Figure 4c utilise l'attribut `@ref` pour dire que l'occurence du mot *chérie* fait référence à une personne à laquelle a été donnée, dans les métadonnées du document TEI, l'identifiant `"Kelly"`. [3]

2.1 Qu'est-ce qu'est la TEI ?

Comme montrent les exemples de la Figure 5, le schème TEI se spécialise à la représentation d'un texte et à son édition numérique. Il facilite l'enrichissement du texte avec les métadonnées, telles que les références aux autres endroits dans le document ainsi que la classification de la nature d'un mot ou d'une phrase. Les normes de la TEI sont souples à exprès, afin de permettre les encodages personnalisés qui se focalisent sur les aspects différents d'un texte. Le même texte peut donc être encodé en TEI dans plusieurs manières, selon les besoins et les objectifs des personnes qui se charge de l'encodage.

Les normes de la TEI sont maintenues par une communauté internationale et leur usage est très répandu dans le monde. L'association est soutenue par le financement des institutions patrimoniales qui comptent sur ses *guidelines* et contribuent des cas d'utilisation. Sur son site web, l'association explique qu'elle continue à modifier ses normes selon les besoins des utilisateurs.

> The scope of the TEI is constantly expanding and the Guidelines are in steady ongoing development to keep pace with the emerging needs of the TEI community. [4]

3. Dans le TEI, les identifiants n'ont pas de mot-dièse, mais quand ils sont référencés dans le document la référence en porte un.

4. *About – TEI : Text Encoding Initiative*. URL : `https://tei-c.org/about/` (visité le 25/08/2022).

La croissance de la TEI rend le schème très approprié à l'édition et à la diffusion des textes puisque beaucoup d'institutions ont développé des outils numériques qui l'utilisent.

La souplesse de la TEI est à la fois un avantage et un défi à surmonter. Puisque le schème permet de plusieurs encodages du même document, il est donc possible de réaliser plusieurs transformations d'un encodage en ALTO vers un encodage en TEI. Mais pour mettre en œuvre une transformation automatique à l'échelle, il faut une seule modélisation qui s'adapte à tout type de document dont la transcription est encodée en ALTO. En outre, l'enrichissement du texte possible dans la TEI est compliqué à réaliser par ordinateur. Tandis qu'un humain pourrait voir la date en-tête sur la lettre dans la Figure 4a et puis savoir que la date référencée dans la salutation est le jour suivant, le 13 août, un logiciel ne pourrait pas faire le liaison entre les deux données si facilement. Donc, bien qu'il puisse savoir, grâce au TAL, que le mot *demain* veut parler d'une date, il ne saurait pas de quelle date parle la lettre ; par contre, une lectrice ou un lecteur humain la saurait avec facilité. Voici quelques défis d'une transformation d'ALTO à TEI.

2.2 Les éléments de base de la TEI

La TEI peut s'adapter à plusieurs types de documents mais elle exige toujours certains éléments de la racine qui donnent au schème son arborescence générale. Depuis la racine `<TEI>` d'un document TEI, il faut au moins ces deux descendants : le `<teiHeader>` et le `<body>`. Comme le schème ALTO, le schème TEI a besoin des métadonnées à propos du document encodé et l'encodage lui-même. Le document TEI imbrique les métadonnées dans l'élément `<teiHeader>`. L'élément `<body>` porte sur les données qui constituent la transcription ou la représentation du document ou des documents ; le dernier sera le cas où le document TEI réalise une édition critique qui ressemblent plusieurs exemplaires d'une œuvre, par exemple. Pour résumer, la TEI a besoin d'au moins les métadonnées, encodées dans le `<teiHeader>`, et les données qui représentent le texte, encodées dans le `<body>`.

Après ces deux éléments obligatoires, la TEI permet aux autres éléments facultatifs de descendre directement de la racine `<TEI>`. L'un d'eux est l'élément `<sourceDoc>` dont nous parlons dans la section **??**.

Expliquer qu'il y a deux éléments essentiels de la racine, le `<teiHeader>` et le `<body>`. Ensuite expliquer l'utilité de l'élément facultatif `<sourceDoc>` et expliquer pourquoi il convient bien aux données de structure d'un fichier ALTO.