Cahiers GUTenberg

UNE COMPARAISON SGML-XML

Sarra Ben Lagha, Walid Sadfi, Mohammed Ben Ahmed

Cahiers GUTenberg, nº 33-34 (1999), p. 127-154.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1999___33-34_127_0>

Comparaison SGML-XML

Sarra BEN LAGHA, Walid SADFI et Mohamed BEN AHMED

Laboratoire RIADI, École Nationale des Sciences de l'Informatique,
Rue des Entrepreneurs 2035, La Charguia II, Tunis-Carthage.
<bellagha@lith.di.epfl.ch>, <sadfi@lri.fr>

Résumé. Le tapage médiatique autour du langage de balisage extensible (eXtensible Mark-up Language (XML)) donne beaucoup d'espoir à propos de l'avenir du web en matière de structuration et de réutilisation des documents. En essayant d'augmenter la flexibilité du langage HTML sans atteindre la rigidité de la norme SGML, XML est considéré comme un pas en avant dans les améliorations techniques des deux mondes (HTML et SGML). Dans cet article, nous nous intéressons aux différences (ou améliorations) de XML par rapport à la norme SGML. Tout document XML étant par construction conforme à la norme SGML, nous présentons dans ce travail les principes de base des deux standards ainsi qu'une comparaison de XML par rapport à SGML.

Abstract. *The media hype surrounding the eXtensible Markup Language (*XML*) leads us to hope that future Web documents will be better structured and easier to re-use. The* XML *specification, which addresses the wish of the Web community to have a language more flexible than* HTML *without necessarily adopting the rigidity and complexity of* SGML*, is considered a step forward since it incorporates technical advances of both the* HTML *and* SGML *worlds. In the present article we explain the differences (improvements) between* XML *and* SGML*. Since each* XML *document is by construction a valid* SGML *document, we review the basic principles of both standards and present a detailed comparison of* XML *and* SGML*.*

Mot-clés : DTD, HTML, SGML, XML.

1. Introduction

Le langage de balisage extensible (XML) est considéré comme un sous-ensemble du langage standardisé de balisage généralisé (SGML) permettant de faciliter l'utilisation de ce dernier sur le web tout en étant interopérable avec SGML et HTML (Hypertext Markup Language). Cependant, dans son document de spécification, le langage XML n'est pas présenté par rapport à SGML ce qui rend parfois difficile la comparaison directe entre les deux standards.

Nous nous intéressons dans cet article à mettre en relief les points de ressemblance et les différences essentielles entre SGML et XML. Pour cela, nous commençons par

présenter les principes de base des deux standards, puis nous passons à la description détaillée des architectures de documents et de modèles de documents en SGML en présentant pour chaque partie ce qui lui correspond en XML.

2. Principe général

Un document est dit au format SGML si toutes ses composantes logiques sont délimitées par des balises conformément à un *langage de balisage* prédéfini. SGML peut être considéré comme un méta-langage pour la définition de langages de balisage. Un document SGML est balisé, selon un schéma ou langage défini par une Définition de Type de Documents (DTD). Tout document possède sa propre structure logique spécifique tout en étant une instance d'une classe de documents dont la structure logique (générique) est définie par une DTD. SGML offre donc le moyen de définir des standards de balisage que sont les DTD. Les documents sont balisés conformément à une DTD et les DTD sont écrites conformément aux règles de la norme SGML.

Une DTD définit principalement les composants et leurs types de contenu qui sont décrits moyenant des connecteurs et des indicateurs d'occurrences ; au niveaux le plus bas de la hiérarchie, on retrouve des types prédéfinis qui sont généralement des chaînes de caractères. L'exemple suivant représente une DTD SGML décrivant la structure logique générique de la classe de documents `chapitre` ainsi que son utilisation pour le balisage d'une instance de cette classe.

```
<!DOCTYPE chapitre [
<!ELEMENT chapitre     - - (titre, préambule?, section+)>
<!ELEMENT titre        - o (#PCDATA)>
<!ELEMENT préambule    - - (paragraphe+)>
<!ELEMENT section      - o (titresection, paragraphe+, soussection*)>
<!ELEMENT titresection - o (#PCDATA)>
<!ELEMENT paragraphe   - o (#PCDATA)>
<!ELEMENT soussection  - o (titrssection, paragraphe+)>
<!ELEMENT titrssection - o (#PCDATA)>
]>
<chapitre>
<titre>SGML : un tour d'horizon</titre>
<section>
<titresection>Une introduction à SGML
<soussection>
<titrssection>Qu'est ce que SGML ?
<paragraphe>
Lorsqu'on voit la panoplie des outils logiciels et...
<paragraphe>
Tout document peut en réalité être considéré sous trois aspects ...
<paragraphe>
```

```
Dans toute application informatique, il est ...
</section>
<section>
<titresection>Un bref historique
<soussection>
<titrssection>Les origines de SGML
<paragraphe>
  ...
<soussection>
<titrssection>...
<paragraphe>
  ...
</section>
</chapitre>
```

Dans l'instance de document, le contenu de tout élément logique est encadré par deux balises : une *balise de début ou d'ouverture* de la forme : `<balise>` et une *balise de fin ou de fermeture*, de la forme : `</balise>`. Les identificateurs des balises et leur mode d'utilisation dans les instances de documents sont spécifiés par la DTD et par la déclaration SGML que nous verrons plus loin.

Le langage de balisage extensible (XML) utilise les mêmes principes de la norme SGML en distinguant toutefois deux catégories de documents : les documents XML *bien formés* et les documents XML *valides*. Tout document XML est formé d'objets textuels appelés entités correspondant à du contenu et du balisage. Une partie de contenu correctement délimitée par une balise de début et une balise de fin correspond à un élément. Tout document XML bien formé correspond à un élément *racine* appelé aussi *élément document* contenant un ou plusieurs éléments dont le contenu est formé par des éléments correctement imbriqués les uns dans les autres. Tout document XML doit être bien formé. Si de plus tous les éléments de son contenu sont déclarés dans une DTD en se limitant au sous-ensemble de fonctionnalités de SGML autorisé par XML, le document en question est dit un *document valide*. Ainsi, tout document XML valide est, par construction, un document conforme à SGML.

3. Syntaxes de déclaration et de balisage

Le nom « SGML » est une abréviation de *Standard Generalized Markup Language* ce qui signifie « Langage normalisé de balisage généralisé » et donne l'impression que SGML est un « langage de balisage » ; de même pour le nom « XML » traduit par « langage de balisage extensible ». Cependant, les documents SGML et XML sont des documents dont les contenus sont balisés conformément à des schémas ou langages de balisage qui sont les DTD. SGML et XML ne définissent donc aucun langage de balisage, mais offrent le moyen d'en définir une infinité selon les besoins de chacun ou plutôt de chaque groupe d'utilisateurs.

Pour écrire une DTD, il faut disposer d'une syntaxe. Pour SGML *la syntaxe elle-même (utilisée pour écrire une* DTD*) peut être propre à une application ou à un système donné*. Ainsi, deux DTD conformes à la norme SGML peuvent ne pas utiliser les mêmes symboles (délimiteurs, mots-clés, etc.) et deux instances conformes à ces DTD n'utiliseront pas le même balisage.

À titre d'exemple, la chaîne « `<!` » indique le début d'une déclaration de balisage. Ceci n'est cependant pas toujours vrai puisque le langage SGML est défini de manière beaucoup plus abstraite. Ainsi, chaque concept est associé à une variable syntaxique. À titre d'exemple, l'ouverture d'une déclaration de balisage est définie par `MDO` *(Markup Declaration Open delimiter)*, un indicateur d'occurrence optionnelle est défini par la variable `OPT` *(OPTional occurrence indicator)*, et ainsi de suite.

L'ensemble des variables syntaxiques et de leur modes d'utilisation forment la définition de la *syntaxe abstraite* du langage SGML. L'utilisation de SGML dans une quelconque application nécessite la traduction des notions abstraites dans un langage concret. La définition d'une *syntaxe concrète* du langage SGML consiste en l'assignation de symboles précis aux variables syntaxiques. Ainsi, la chaîne « `<!` » peut, par exemple, être utilisée comme délimiteur d'ouverture de déclaration de balisage, elle est alors associée à la variable `MDO`. De même, la variable `OPT` peut être représentée par le symbole « ? », etc.. Plusieurs syntaxes peuvent donc être associées au langage de balisage SGML. Par ailleurs, pour faciliter et garantir l'échange des documents SGML entre plate-formes différentes, tout système SGML doit au moins supporter une syntaxe concrète appelée *syntaxe concrète de référence*. La syntaxe concrète de référence est définie et utilisée par la norme elle-même. Les risques d'erreur et d'ambiguïté n'étant pas nuls, cette possiblité de redéfinition de la syntaxe concrète a été jugée inutile par les concepteurs du standard XML.

Pour éviter qu'un **système SGML soit incapable de gérer un document conforme à une DTD SGML correcte** (par rapport aux spécifications de la norme), **tout système SGML définissant sa propre syntaxe concrète, doit disposer d'une application de transcription vers la syntaxe concrète de référence** pour une garantie de compatibilité entre systèmes hétérogènes. **En XML, la seule syntaxe autorisée est la syntaxe concrète de référence** utilisée dans le document de spécification du standard (cette syntaxe est légèrement différentes de la syntaxe concrète de référence de la norme SGML comme nous le montrerons au fur et à mesure).

4. Architecture des documents

Comme nous l'avons dit plus haut, tout document SGML contient une DTD et un contenu balisé conformément à cette DTD. Mais en réalité, l'architecture d'un document SGML est légèrement plus compliquée. En effet, un document SGML contient trois grandes parties : la *déclaration* SGML, le *prologue* et l'*instance balisée*. On re-

trouve le cas le plus simple, lorsque la déclaration contient l'ensemble des caractères utilisés dans la DTD (qui peut être prédéfini) et que le prologue contient une DTD.

Les documents XML sont par contre formés de deux parties uniquement : le *prologue* qui contient aussi bien la déclaration XML que la DTD, et l'*élément document* qui correspond à l'instance balisée.

4.1. Déclarations SGML et XML

La déclaration SGML définit toutes les informations nécessaires à l'analyse d'un document par une application. Elle indique principalement le jeu de caractères utilisé (`CHARSET`, `BASESET`, `DESCSET`), l'ensemble des capacités (`CAPACITY`) requises par le système pour analyser le document, la portée de la syntaxe concrète (`SCOPE`) et les modifications éventuelles de cette syntaxe (`SYNTAX`). La déclaration SGML définit aussi un certain nombre de fonctionnalités optionnelles (`FEATURES`) destinées à réduire le balisage ou à rattacher des traitements automatiques à certaines parties des documents, ainsi que d'éventuelles informations (`APPINFO`) spécifiques à l'application qui analysera et traitera les documents.

Quant à la déclaration XML, elle indique uniquement la version du standard XML utilisé dans le document.

EXEMPLE

Une déclaration SGML :

```
<!SGML "ISO 8879:1986"
   CHARSET
         BASESET "ISO 646:1983 //CHARSET
                  International reference Version (IRV)// ESC 2/5 4/0"
         DESCSET 0   9 UNUSED
                  127 1 UNUSED
   CAPACITY PUBLIC "ISO 8879:1986//CAPACITY Reference //EN"
   SCOPE    DOCUMENT
   SYNTAX   PUBLIC "ISO 8879:1986//SYNTAX Reference //EN"
   FEATURES
     MINIMIZE DATATAG NO OMITTAG  YES RANK     NO  SHORTTAG  YES
     LINK     SIMPLE  NO IMPLICIT NO  EXPLICIT NO
     OTHER    CONCUR  NO SUBDOC   NO  FORMAL   NO
   APPINFO NONE
>
```

Une déclaration XML :

```
<?xml version="1.0"?>
```

La déclaration SGML décrit les fonctionnalités de la norme autorisées ou interdites dans une classe de documents. Par construction, tout document XML est un docu-

ment SGML particulier qui n'utilise que les fonctionnalités de SGML autorisées par les spécifications du standard XML. L'indication de la version de XML utilisée dans le document suffit ainsi à déterminer les fonctionnalités de SGML permises dans le document XML en question. La déclaration XML est donc une déclaration SGML implicite, induite par le numéro de la version utilisée de XML. Il est à noter qu'à ce jour, la seule version qui existe de XML est la version 1.0 mais de futures versions ne sont pas exclues par le groupe de travail XML.

Dans ce qui suit, nous décrivons les différentes parties d'une déclaration SGML et leurs correspondants dans les documents XML.

4.1.1. Jeu de caractères

Pour garantir la portabilité des documents SGML entre systèmes hétérogènes, une indication explicite du jeu de caractères utilisé doit être mentionnée dans la déclaration SGML. Pour que le document puisse être traité, le jeu de caractères du document doit être le même que le jeu de caractères du système. Si tel n'est pas le cas, l'un ou l'autre doit être modifié.

La déclaration du jeu de caractères se fait par le mot clé `CHARSET`, qui doit être suivi d'une indication du jeu de caractères de base (généralement une référence à un jeu de caractères normalisé) indiqué par le mot clé `BASESET`, et les modifications éventuelles indiquées par `DESCSET`.

EXEMPLE

Soit la déclaration suivante :

```
CHARSET
   BASESET "ISO 646:1983 //CHARSET
           International reference Version (IRV)// ESC 2/5 4/0"
   DESCSET 0   9 UNUSED
           127 1 UNUSED
```

Ceci signifie que le jeu de caractères de base est le jeu défini par la norme « ISO 646:1983 ». Les neufs caractères commençant à 0 ne sont pas utilisés ni le caractère 127.

Afin de déduire le jeu de caractères utilisé dans les documents, il faut déjà que le système comprenne la déclaration SGML elle même (où sont déclarés les caractères utilisés) ; la déclaration SGML doit pour celà n'utiliser que le codage ASCII.

En XML, les caractères sont codés essentiellement en `UTF-8` et `UTF-16`. Tout document utilisant un autre codage doit commencer par une déclaration de codage dans la déclaration XML, comme le montrent les exemples suivants :

```
<?xml encoding="UTF-8"?>
<?xml encoding="EUC-JP"?>
```

À la différence avec SGML, la déclaration de codage XML tient le rôle d'étiquette interne à chaque entité, indiquant quel codage de caractères est utilisé. L'étiquette elle-même doit cependant être compréhensible par le système ou l'application XML ce qui nécessite une information sur le codage ! La situation n'est tout de même pas sans issue puisque, d'une part, toute application XML gère un nombre fini de codages de caractères et, d'autre part, XML impose des restrictions sur la position et le contenu de la déclaration de codage XML de sorte qu'une détection automatique du codage dans chaque entité devienne possible dans la majorité des cas, sachant que d'autres sources d'information peuvent aussi être fournies avec les documents pour éclairer les types de codage utilisés.

4.1.2. Ensemble de capacités

Les capacités (`CAPACITY`) sont des mesures de la mémoire requise pour stocker l'analyse d'une DTD. Elles sont aussi utilisées pour définir le minimum qu'un système SGML doit être capable d'analyser. Les capacités sont exprimées en *unités de capacité* (*capacity point*). Une unité de capacité correspond à l'espace mémoire nécessaire pour le stockage d'un caractère.

Dans la syntaxe SGML sont définis des compteurs tels que `ENTCAP` pour les entités, `ATTCAP` pour les attributs, etc. La capacité de chaque objet est calculée en multipliant la valeur du compteur (définie dans la syntaxe) par le nombre d'occurrences de l'objet en question de la DTD. La somme de toutes ces valeurs ne doit pas dépasser la capacité totale fixée pour une DTD.

La norme SGML définit un jeu de capacités de référence [1] qui correspond à 35 000 caractères. De nos jours, cette capacité est jugée assez faible. L'espace mémoire n'étant plus un problème pour les configurations des systèmes actuels, le standard XML n'impose aucune contrainte sur l'espace nécessaire à l'analyse et au stockage des DTD. Les capacités sont illimitées pour les documents XML.

4.1.3. Portée de la syntaxe

Comme nous l'avons signalé plus haut, tout document SGML est composé de trois parties : la déclaration SGML, le prologue et l'instance balisée. Si un document ou une application SGML définit sa propre syntaxe concrète (appelée variante de la syntaxe de référence), elle ne peut l'utiliser que dans l'instance balisée et le prologue. De telles syntaxes sont définies dans la déclaration SGML en utilisant forcément une syntaxe qui doit être compréhensible par tout système SGML et qui ne peut donc être que la syntaxe concrète de référence. Lorsqu'un document SGML définit sa propre syntaxe, le paramètre (`SCOPE`) de la déclaration SGML indique respectivement par l'une des

1. Des valeurs prédéfinies sont associées aux différents compteurs par la syntaxe concrète de référence.

deux valeurs (DOCUMENT) ou (INSTANCE) si la syntaxe définie s'applique à tout le reste du document (la DTD et l'instance balisée) ou à l'instance balisée uniquement.

Dans les documents XML cette notion de portée de la syntaxe (SCOPE) n'a pas raison d'être puisque toutes les applications et les documents XML sont conformes à une seule syntaxe (la syntaxe XML) qui peut elle-même être considérée comme une variante de la syntaxe concrète de référence de la norme SGML. En d'autres termes, le standard XML ne définit aucun paramètre identique ou équivalent à (SCOPE), mais lorsqu'on définit une classe de documents SGML reprenant les contraintes imposées par XML, la déclaration SGML correspondante donne la valeur (DOCUMENT) au paramètre (SCOPE).

4.1.4. Syntaxes concrètes

La syntaxe utilisée dans tout document et toute DTD XML correspond à la syntaxe concrète de référence de la norme SGML moyennant quelques différences. Nous renvoyons le lecteur intéressé par les deux syntaxes aux documents de spécifications des deux standards et nous nous limitons ici à signaler les principales différences syntaxiques de XML par rapport à SGML, à savoir, dans XML :

- Le délimiteur de fermeture d'une déclaration d'instruction de traitement (PIC : *Processing Instruction Close*) correspond aux deux caractères ?> plutôt qu'au caractère >.
- Les lettres miniscules et majuscules sont considérées comme différentes dans les identificateurs de nom (NAME).
- Un identificateur de nom (NAME) peut utiliser les deux caractères _ et , en plus des lettres et des chiffres.
- Les identificateurs de noms peuvent être définis en Unicode et ne sont donc pas restreints aux caractères de l'ASCII.
- Les identificateurs commançant par [Xx][Mm][Ll] sont réservés.
- Deux attributs sont prédéfinis en XML. Il s'agit de xml:space et xml:lang qui sont utilisés par les processeurs XML pour gérer respectivement les espaces et les langues.
- Des entités caractères : &, <, >, ', " sont prédéfinies pour les caractères &, <, >, ' et " respectivement.

4.1.5. Fonctionnalités optionnelles

Les fonctionnalités optionnelles offertes par SGML sont au nombre de onze et sont regroupées en trois catégories appelées : *minimisation de balisage* (MINIMIZE), *utilisation des liens* (LINK) et quelques *autres* fonctionnalités (OTHER) qui concernent essentiellement l'extension des types de documents. La majorité de ces fonctionnalités a pour objectif de minimiser le nombre de caractères saisis lors du balisage (supposé

manuel). Ceci donne lieu à des documents balisés dont la structure est peu compréhensible par un lecteur humain. De plus, l'utilisation de ces options nécessite beaucoup plus d'expertise que la rédaction des DTD ou le balisage des documents. Pour ces raisons, la quasi-totalité de ces fonctionnalités a été interdite par le standard XML.

L'utilisation des fonctionnalités optionnelles est indiquée par un paramètre dans la déclaration SGML. Ainsi, pour chacune des fonctionnalités, il faut indiquer si « oui » (`yes`) ou « non » (`no`) elle est utilisée comme le montre l'exemple suivant où seules les fonctionnalités `shorttag` et `omittag` sont activées :

```
features
  minimize datatag no omittag  yes rank     no  shorttag  yes
  link     simple  no implicit no  explicit no
  other    concur  no subdoc   no  formal   no
```

Dans ce qui suit, nous reprenons une par une l'ensemble de ces fonctionnalités optionnelles.

Fonctionnalités de minimisation de balisage

Toute instance de document SGML est essentiellement formée par le contenu et le balisage. Dans plusieurs situations, certaines informations de balisage peuvent être déduite à partir du contexte de l'instance. Les fonctionnalités de minimisation de balisage offrent le moyen d'exploiter le contexte en autorisant l'abréviation de balises et d'appels d'entités, ainsi que la suppression de certaines balises.

La minimisation de balisage n'affecte que l'instance du document. Pour être transmises vers un système SGML ne supportant pas les minimisations, les instances de documents réduites doivent donc passer par un traitement de rétablissement de balisage.

La minimisation du balisage est définie par les fonctionnalités suivantes :

`OMITTAG` signifie que l'on peut omettre certaines balises (de début ou de fin) lorsque celles-ci peuvent être déduites d'après le contexte.

EXEMPLE

Soit la déclaration de balisage SGML suivante :

```
<!ELEMENT Liste  - -  (item)+>
<!ELEMENT item   - -  (#PCDATA)>
```

Dans une instance de document balisée, on retrouve par exemple :

```
<Liste>
<item>élément 1
</item>
<item>élément 2
```

```
</item>
<item>élément 3
</item>
 ...
</Liste>
```

Le début d'un nouvel élément de liste implique forcément la fin de l'élément précédent. Ainsi, une minimisation par omission de balisage consiste à rendre obligatoire uniquement la balise de début de chaque élément, ce qui donne le texte suivant :

```
<Liste>
<item>élément 1
<item>élément 2
<item>élément 3
  ...
</Liste>
```

OMITTAG est une fonctionnalité interdite dans XML. Tout document XML devant être bien formé, chaque élément de son contenu doit obligatoirement être encadré par une balise de début et une balise de fin, ces dernières ne sont ainsi jamais omissibles.

SHORTTAG signifie que l'on peut abréger des balises en omettant des délimiteurs, des identificateurs génériques ou des spécifications d'attributs. Ceci donne lieu respectivement à des balises abrégées non fermées, des balises vides ou des minimisations d'attributs.

- Le délimiteur de fin de balise peut être omis lorsque celle-ci est immédiatement suivie par une autre.
  EXEMPLE
  Le contenu suivant :
  ```
  <Liste><item>élément 1</item></Liste>
  ```
  peut être abrégé comme suit :
  ```
  <Liste<item>élément 1</item</Liste>
  ```
  Les balises abrégées non fermées sont interdites dans XML (même si on spécifie SHORTTAG yes).
- Une balise abrégée vide est une balise dans laquelle aucun identificateur générique (IG) ni attribut n'est spécifié. L'analyseur suppose dans ce cas que l'IG est le même que celui du dernier élément ouvert.
  EXEMPLE
  La liste de l'exemple précédent peut être abrégée comme suit :
  ```
  <Liste>
  <item>élément 1</>
  <>élément 2</>
  </Liste>
  ```
  Le balisage peut être davantage réduit en utilisant le délimiteur composé d'un caractère unique normalement / il s'agit alors d'une « balise de fin

nulle » (NET) à condition que le même caractère ait été utilisé comme délimiteur de fin de balise.

EXEMPLE

Dans le balisage suivant, les balises de fin sont vides et représentée par le caractère / qui est par ailleurs utilisé comme un caractère de données (dans les chaînes 1/2 et 2/2) :

```
<Liste>
 <item/ élément 1/2/
 <item/ élément 2/2/
</Liste>
```

Les balises abrégées vides sont interdites dans XML. Les balises à fin nulles (NET) sont autorisées uniquement lorsque l'élément balisé est un élément de contenu vide (EMPTY). Le délimiteur de fin nulle (NET) est représenté par les deux caractères />. Ainsi, XML n'autorise pas les balises de fermeture (ou de fin) à fin nulles mais autorise des balises de début à fin nulles pour les éléments vides qui n'ont donc pas de balise de fermeture (ou de fin).

EXEMPLE

Le balisage XML suivant :

```
<Liste>
 <item><image/></item>
 <item>Un texte ...</item>
</Liste>
```

signifie que l'élément image est un élément de type EMPTY qui possède une balise de début et pas de balise de fin et dont le délimiteur de fin de cette balise est nul (NET) c'est à dire />.

– L'abréviation par minimisation d'attribut consiste en l'élimination totale ou partielle d'une liste de spécification d'attributs. Selon les situations, il peut s'agir d'enlever les délimiteurs de valeur (apostrophes ou guillemets), de la valeur de l'attribut (valeur par défaut ou impliquée par le système), de l'identificateur de l'attribut (en indiquant uniquement la valeur), etc. Le premier cas et le dernier sont interdits par XML même lorsque la fonctionnalité SHORTTAG est autorisée.

SHORTREF signifie que l'on peut utiliser des délimiteurs d'appels abrégés plutôt que des appels d'entités complets. En d'autres termes, au lieu d'être saisies explicitement, certaines balises peuvent être remplacées par la saisie d'autres caractères ou chaînes, comme les caractères de fonction non imprimables pour un balisage moins visible, l'utilisation des guillemets commes balises, etc. Les délimiteurs sont prédéfinis dans la syntaxe (concrète ou autre) et sont associés à des entités dans une table appelée « table de correspondance d'appels abrégés ». Les appels abrégés ont été conçus pour permettre l'utilisation de balises SGML tout en concervant les conventions habituelles de saisie en traitement de texte plutôt qu'en saisissant explicitement un balisage.

EXEMPLE
Soit les déclarations suivantes[2] :

```
<!ENTITY    ptag    STARTTAG "p">
<!SHORTREF  wisiwyg "&#TAB;"     ptag
                    "&#RS;&#RE;" ptag>
```

Cela signifie qu'une balise de début d'un élément paragraphe p, peut être induite (comme si l'on avait saisi la balise <p>) par la saisie d'un taquet de tabulation (TAB) ou par la saisie de la séquence fin de ligne et retour à la ligne (`line feed` et `carriage return`). La fonctionnalité SHORTREF est interdite dans XML.

DATATAG signifie que certaines données peuvent être interprétées à la fois comme des balises et des données.
EXEMPLE
Soit la déclaration d'élément suivante :

```
<!ELEMENT NomComplet - -  ([ nom, ", ", "  "] , prenom)>
<!ELEMENT nom        - -  (#PCDATA)>
<!ELEMENT prenom     - -  (#PCDATA)>
```

Cela signifie que l'identificateur générique de l'élément réductible est nom, que le modèle de balise textuelle est , (virgule suivie d'un espace) et que le modèle de remplissage de balises textuelles est (deux espaces).
Cette déclaration de balisage signifie que toute chaîne formée par une virgule suivie d'espaces (un ou plusieurs) apparaissant après un élément nom (précédé par la balise <nom>) dans un élément NomComplet est considérée comme balise de fin de nom et comme partie des données de l'élément NomComplet. La fonctionnalité DATATAG est interdite par XML.

RANK signifie que l'on peut omettre des indicateurs de niveaux d'imbrication. SGML offre la possibilité de déclarer des éléments hiérarchisés. Il s'agit d'éléments imbriqués avec le même type de contenu. Un élément hiérarchisé possède un identificateur générique (IG) formé par un *radical de niveau* et un *suffixe de niveau*. La fonctionnalité RANK permet d'omettre les indicateurs de niveaux (les suffixes de niveaux) qui figurent normalement dans les balises.
EXEMPLE
Soit la déclaration de balisage SGML suivante :

```
<!ELEMENT parag 1 - o (#PCDATA, parag2*)>
<!ELEMENT parag 2 - o (#PCDATA, parag3*)>
<!ELEMENT parag 3 - o (#PCDATA)>
```

2. Cet exemple est repris de l'annexe F de la norme SGML.

L'élément parag est hiérarchisé. Cette déclaration peut donner lieu au balisage suivant :

```
<parag1>  un texte de niveau 1                </parag1>
<parag1>  un autre texte de niveau 1          </parag1>
<parag2>  un texte de niveau 2                </parag2>
<parag2>  un autre texte de niveau 2          </parag2>
<parag1>  de nouveau, un texte de niveau 1    </parag1>
```

La minimisation du balisage par la fonctionnalité RANK consiste à déduire l'IG complet à partir du radical et du dernier suffixe utilisé, ce qui donne lieu au balisage suivant (où l'on utilise aussi une minimisation par omission des balises de fin) :

```
<parag1>  un texte de niveau 1
<parag>   un autre texte de niveau 1
<parag2>  un texte de niveau 2
<parag>   un autre texte de niveau 2
<parag1>  de nouveau, un texte de niveau 1
```

La fonctionnalité RANK est interdite par XML.

Fonctionnalités de type de lien

L'un des principaux apports de la structuration des documents est la possibilité de leur associer des traitements automatiques *a priori*, c'est-à-dire avant d'en disposer concrètement. L'utilisation des fonctionnalités LINK permet d'associer plusieurs structures à un même document. Ceci dans le but de définir à titre d'exemple la structure d'un document avant et après l'application d'un traitement particulier.

La multiplicité des structures pour un même type de document peut être faite par : les définitions de processus de lien qui permettent de définir la façon dont un document peut être transformé d'un type en un autre (comment générer un balisage à partir d'un autre), ou par les instances de documents concurrentes où les balisages correspondent à des instances de différents types de documents.

Une déclaration de type de lien est identifiée par un nom et précise le type de document source ainsi que le type de document résultant. Plusieurs type de liens pouvant être utilisés simultannément, on parle alors de chaîne de processus de lien. La spécification d'un type de lien peut être :

SIMPLE lorsque le type de document source est le type de document de base.
IMPLICIT si le nom du type de document résultant est déduit par l'application.
EXPLICIT quand le nom de type de document source doit être le type de documents de base, ou un autre type de document résultant dans une chaîne de processus.

Les trois fonctionnalités de type de lien SIMPLE, IMPLICIT et EXPLICIT sont interdites dans XML.

Autres fonctionnalités

Ce sont des fonctionnalités qui essayent de prendre en considération la possibilité de gérer plusieurs vues d'un même document, la possibilité d'utiliser plusieurs types de sous-documents dans un même document, ou encore de formaliser les identificateurs de textes publics. Il s'agit respectivement des trois fonctionnalités suivantes :

`CONCUR` signifie que plusieurs balisages peuvent être utilisés au sein d'un même document. Les balises provenant d'autre types de documents (autres que le type de document de base) contiennent alors des préfixes indiquant le type de document où ils sont définis. Ainsi, aucune confusion ne risque d'être faite entre les différentes balises.

`SUBDOC` signifie que certaines parties d'une DTD peuvent avoir leur propres environnements. Tout document SGML est formé d'entités. Celle qui contient la définition de type de document, la déclaration SGML, etc. est appelée *entité document* SGML. L'utilisation de la fonctionnalité `SUBDOC` permet à des entités SGML de se comporter comme des entités document SGML, c'est à dire de faire référence à leur propre environnement de définition de balisage. Les types de documents et les déclarations d'entités de l'entité appelant un sous-document, sont suspendus quand l'entité sous-document est ouverte, et reprennent quand elle se termine.

`FORMAL` signifie que les identificateurs publics ont une structure formelle qui possède un certain nombre de composantes (tels que identificateurs de texte public, une désignation de la langue utilisée, etc.).

Ces trois fonctionnalités sont interdites dans XML.

4.1.6. Informations spécifiques

Si pour traiter un document dans un système SGML il est nécessaire de disposer d'*informations spécifiques à l'application* (*application specific information*), celles-ci s'insèrent après le mot clé `APPINFO` dans la déclaration SGML. Dans le cas contraire, le mot clé `NONE` est utilisé pour indiquer qu'aucune information spécifique n'est utilisée. C'est le cas de tous les documents XML.

4.2. Le prologue

Le prologue dans SGML définit le type de document utilisé. Théoriquement, il s'agit d'une DTD, mais tel n'est pas le cas si les fonctionnalités `CONCUR` ou `LINK` sont utilisées dans la déclaration SGML.

La DTD peut être totalement définie dans le prologue (toutes les déclarations d'éléments, d'attributs, d'entités et du contenu des fonctionnalités déclarées plus haut

dans la déclaration SGML). Cependant, elle peut être totalement définie comme entité dans un fichier externe et référencée dans le prologue ou encore être formée d'une partie sous la forme d'un appel à une entité externe et d'un ensemble de déclarations de balisage.

Dans les documents XML, le prologue contient aussi la déclaration XML. De même que dans SGML, les DTD XML peuvent être incluses et/ou référencées dans le prologue qui correspond donc aux DTD.

4.3. L'instance balisée

C'est dans l'instance de document balisée que nous retrouvons le contenu réel du document « SGML isé » ou « bien formé ». Une instance est toujours conforme à une DTD (ou à plusieurs) qui doit la précéder[3]. Dans les instances de documents SGML, la présence et l'absence des balises sont conditionnées par les déclarations faites dans la DTD. Dans les documents XML, toutes les balises sont obligatoires ; leur sémantique et leur ordre doit être conforme aux déclarations de la DTD.

5. Architecture de DTD

Normaliser les documents électroniques selon SGML ou XML, consiste à doter chaque document des informations concernant les types et l'organisation de ses différents composants. Pour une classe de documents, ces informations sont indiquées par des balises définies par un langage de balisage qui est la DTD.

Une DTD définit en fait trois types de commandes de balisage : les *éléments*, les *attributs* et les *entités*. Elle peut aussi contenir des déclarations d'*instructions de traitement*, et offre comme tout langage, le moyen de *commenter* les différentes déclarations. De plus, la DTD offre le moyen de définir des parties du contenu informationnel des documents balisés non destinées à une interprétation par l'analyseur. Il s'agit des *sections marquées* et des contenus conformes à des *notations* particulières. Dans ce qui suit, nous présentons, plus en détails, chacune de ces notions dans les deux standards.

5.1. Déclaration d'éléments

Une déclaration d'élément en SGML est composée des parties suivantes :

- un délimiteur d'ouverture de déclaration de balisage,
- un mot-clé indiquant le type de la déclaration,
- l'identificateur générique de l'élément,

3. La DTD est définie ou référencée dans le prologue du document.

– une indication de minimisation éventuelle de balisage par omission,
– le contenu de l'élément,
– un délimiteur de fermeture de déclaration de balisage.

Les éléments en XML sont déclarés de la même manière à la seule différence qu'ils ne contiennent pas d'indication de minimisation de balisage.

Selon la syntaxe concrète de référence, les *délimiteurs d'ouverture et de fermeture de déclarations de balisage* correspondent respectivement à la chaîne : `<!` et au symbole : `>`. Le *mot-clé* indiquant la déclaration d'un élément est : `ELEMENT`. L'utilisateur définit l'*identificateur générique* de l'élément en respectant certaines contraintes (unicité et taille essentiellement).

EXEMPLE

La déclaration d'un élément `X` dont le contenu est de type `Y`, s'écrit en SGML :

```
<!ELEMENT X   - - (Y) >
```

et en XML :

```
<!ELEMENT X (Y) >
```

5.1.1. Minimisation de balisage par omission (dans SGML*)*

La minimisation de balisage par omission dans SGML est indiquée par deux symboles successifs qui peuvent prendre chacun la valeur – ou o. Le premier symbole est associé à la balise de début qui sera placée avant l'occurrence de l'élément déclaré dans l'instance du document. La seconde correspond à la balise de fin qui sera placée juste après l'élément en question. La valeur de chacun de ces symboles est – si la balise correspondante est obligatoire, et o si la balise est omissible. Cette dernière peut, dans ce cas, être déduite du contexte général de balisage du document. Ainsi, quand l'une des deux balises (de début ou de fin) peut être omise, le signe – qui lui correspond dans la déclaration de l'élément, est remplacé par le signe o. Nous écrivons alors : « – – » si les deux balises sont obligatoires, « o – » si la balise de début est omissible, « – o » si seule la balise de début est obligatoire et enfin « o o » si les deux balises sont omissibles.

5.1.2. Contenu d'un élément

En SGML, tout élément peut contenir des sous-éléments qui peuvent être *propres* ou *inclus*. L'inclusion est une forme d'exception aux types de contenu, que nous détaillons ci-dessous. En XML, les exceptions ne sont pas autorisées.

Exceptions aux modèles de contenu

La DTD définit la structuration hiérarchique des types de contenu. Certains composants (notamment les éléments flottants tels que les notes de bas de page) nécessitent d'autres mécanismes. SGML offre le moyen de définir de telles structures par la notion d'exception.

Soit la définition d'élément suivante :

```
<!ELEMENT  X - - (Y)  >
```

Si un élément `Z` peut apparaître dans tous les sous-éléments de `X`, ceci est effectué directement par une « inclusion » des éléments qui ne font pas partie de la structure hiérarchique. Nous écrivons alors :

```
<!ELEMENT  X - - (Y)+(Z)  >
```

Ce qui signifie que `Z` (qui est une inclusion) peut apparaître un nombre illimité de fois dans n'importe quel sous-élément de `X`. Si par contre, un élément `T` n'est pas obligatoire dans les sous-éléments de `X`, mais il peut y apparaître, et que l'on veut empêcher cela, `T` est alors *exclu* de `X` de la manière suivante :

```
<!ELEMENT  X - - (Y)-(T)  >
```

Cela signifie que `T` existe dans la déclaration de l'élément `Y` comme dans le cas suivant :

```
<!ELEMENT  X - - (Y)-(T)  >
<!ELEMENT  Y - - (S | T)  >
```

EXEMPLE

Si un paragraphe (`Para`) contient du texte (`Texte`) et des notes de bas de pages (`FN`), et que dans le contenu d'une note de bas de page ne doit apparaître aucune note de bas de page au niveau le plus haut d'une autre note, mais qu'une note peut apparaître dans un paragraphe contenu dans une note, cela s'écrit :

```
<!ELEMENT FN    - - (Para+)-(FN)  >
<!ELEMENT Para  - o (Texte | FN)+ >
<!ELEMENT Texte o o (#PCDATA)     >
```

Les inclusions et les exclusions sont appelées des *exceptions*. Une déclaration d'élément en SGML peut contenir des exceptions. Elle a donc l'une des trois formes suivantes :

```
<!ELEMENT Identificateur minimisation Contenu >
<!ELEMENT Identificateur minimisation (Contenu)+(inclusion) >
<!ELEMENT Identificateur minimisation (Contenu)-(exclusion) >
```

En XML, les minimisations de balisage et les exceptions étant interdites, une déclaration d'élément se présente toujours sous la forme suivantes :

```
<!ELEMENT Identificateur Contenu >
```

Contenu (propre) d'un élément

Le contenu de chaque élément est un *modèle de contenu*, ou un *contenu déclaré*, il existe aussi des contenus *mixtes* et des contenus *quelconques*.

Un modèle de contenu spécifie le *groupe modèle* qui définit le contenu autorisé d'un élément. Il est formé par une liste d'éléments de contenu (appelés sous-éléments), séparés par des connecteurs (du même type) suivis chacun par un indicateur d'occurrence, le tout entre deux parenthèses qui correspondent au délimiteur d'ouverture de groupe et au délimiteur de fermeture de groupe respectivement. Si la liste de sous-éléments est réduite à un seul sous-élément, le contenu de l'élément est dit *élémentaire*. Les connecteurs autorisés en SGML sont « , » (qui indique une séquence de sous-éléments de la liste), « & » (un agrégat) et « | » (un choix). En XML, seuls les choix et les séquences sont autorisés (le connecteur & n'est donc pas utilisé).

Un objet de la liste de contenu peut être un identificateur ou une liste d'éléments entre parenthèses. Dans ce cas, les connecteurs reliant ces éléments doivent être du même type, mais ils peuvent être différents des connecteurs de la liste mère. Chaque objet est, par défaut, obligatoire et non répétable. Suivi de l'un des indicateurs d'occurrences suivants : (?), (+) ou (*), il devient respectivement *optionnel*, *obligatoire et répétable* ou *optionnel et répétable*.

Un document SGML ou XML contient du texte, mais il peut contenir des parties nécessitant des traitements particuliers, telles que des figures, des formules, etc. Si le contenu d'un élément n'est pas un texte simple, l'élément aura un *contenu déclaré* (declared content). Un contenu déclaré SGML correspond à l'un des types prédéfinis suivants : CDATA (*Character Data*), RCDATA (*Replaceable Character Data*) ou EMPTY (élément à contenu vide). En XML, les types CDATA et RCDATA sont interdits. Les éléments à contenu vide (dans SGML et XML) ont une balise de début mais pas de balise de fin.

Le contenu le plus élémentaire en SGML est une chaîne de caractères `#PCDATA` (*Parsable Character Data*). Un élément est dit de type « quelconque » s'il peut contenir aussi bien du `#PCDATA` que n'importe quel autre élément défini dans la DTD, dans n'importe quel ordre. Un tel modèle de contenu est défini par le type prédéfini `ANY`. L'utilisation de ce type en SGML est illustrée par l'exemple suivant :

```
<!ELEMENT Paragraphe   - - (#PCDATA | (Titre, Para, Ss-sssection)) >
<!ELEMENT Ss-sssection - - ANY >
```

Cela signifie que la `Ss-sssection` peut elle-même contenir des `Para`, des `Titre`, ou n'importe quel autre type d'élément déclaré dans la DTD SGML.

Dans une version finale d'une DTD valide, on ne doit pas trouver (si l'on veut obtenir des documents structurés !) des éléments de type `ANY`. Ce type de contenu est d'ailleurs interdit en XML.

Quand un élément contient aussi bien des sous-éléments que des chaînes `#PCDATA`, il est appelé élément à « contenu mixte » (*mixed content*). Dans les déclarations d'éléments XML, les modèles de contenus mixtes sont toujours utilisés dans des modèles de contenu correspondnant à des listes de choix répétables. Ainsi, les types des sous-éléments peuvent être contraints mais pas leur ordre ni leur nombre.

5.2. Déclaration d'attributs

Certains des éléments d'un document peuvent avoir des caractéristiques particulières ou doivent être traités d'une certaine façon par l'application SGML ou XML. Ces informations ne faisant pas partie du contenu réel du document, elles ne sont ni des éléments de la DTD, ni des portions balisées dans l'instance du document. Elles sont simplement déclarées comme *attributs* associés à des éléments de la DTD. Les attributs sont utilisés à l'intérieur de la balise de début de l'élément qu'ils décrivent dans l'instance de documents.

EXEMPLE

Soit un restaurant utilisant une application SGML pour générer périodiquement ses menus sous la forme de documents structurés. Dans la rubrique « Desserts » du menu, ce restaurant propose l'élément « Fruits de saison ». Si une telle information doit être suivie par une explication que l'application peut charger directement à partir d'une base de données, une information sur la saison en question serait indispensable à l'établissement de la requête.

L'attribut `Saison` associé à l'élément `fruits` aura ainsi l'une des quatre valeurs possibles `hiver`, `printemps`, `été` et `automne`.

Chaque élément déclaré dans la DTD peut avoir un ou plusieurs attributs. On parle en effet de liste d'attributs pour un élément. Un attribut se compose d'un nom d'at-

tribut, d'un signe = et d'une valeur entre deux délimiteurs de littéraux (ce sont des guillemets ou des apostrophes qui doivent être assortis).

EXEMPLE

Si le restaurant de l'exemple précédant fait partie d'une chaîne internationale, qui utilise la même application SGML ou XML dans toutes ses filiales, la `Saison` seule ne suffit pas pour indiquer quels sont les fruits de la saison ! Un attribut `Pays` serait alors nécessaire pour pouvoir extraire la bonne information. En termes d'instances de documents SGML ou XML, ceci s'écrirait :

```
<fruits Saison="hiver" Pays="Suisse">
    Fruits de saison (pommes...)
</fruits>
```

et dans une autre instance on trouverait :

```
<fruits Saison="hiver" Pays="Tunisie">
    Fruits de saison (oranges...)
</fruits>
```

En SGML, quand l'attribut prend sa valeur dans un ensemble fini de valeurs, son nom peut être omis. Il en est de même pour les délimiteurs de littéraux (" ou ') quand il n'y a pas de blanc dans la valeur de l'attribut. Ainsi :

```
<fruits hiver Suisse>...
```

est équivalent en SGML à :

```
<fruits Saison = "hiver" Pays = "Suisse">...
```

Les attributs et leurs valeurs possibles sont définis dans la DTD par les déclarations de listes de définitions d'attributs *(attribute definition list)* qui comportent les parties suivantes :

- délimiteur d'ouverture de déclaration de balisage : `<!` ;
- mot-clé de déclaration d'une liste de définition d'attributs : `ATTLIST` ;
- nom de l'élément dont l'attribut dépend ;
- énumération des valeurs déclarées de l'attribut ou mot-clé ;
- valeur par défaut de l'attribut ou mot-clé ;
- délimiteur de fermeture de déclaration de balisage : `>`.

EXEMPLE

```
<!Attlist fruits Saison (hiver|pringtemps|été|automne) "hiver" >
<!Attlist fruits Pays   #REQUIRED >
```

En XML, quand plus d'une déclaration de liste d'attributs existe pour un type d'élément donné, le contenu de toutes les déclarations fournies est fusionné. Ceci donne pour l'exemple précédent la déclaration suivante (valable aussi en SGML) :

```
<!ATTLIST fruits Saison (hiver|pringtemps|été|automne) "hiver"
          Pays   #REQUIRED >
```

Les mots clés pour les valeurs déclarées d'attributs en XML sont[4] : `CDATA`, `ENTITY`, `ENTITIES`, `ID`, `IDREF`, `IDREFS`, `NMTOKEN`, `NMTOKENS` et `NOTATION`.

Ces mots clés sont autorisés en SGML qui rajoute en plus les mots clés suivant : `NAME`, `NAMES`, `NUMBER`, `NUMBERS`, `NUTOKEN` et `NUTOKENS`.

Dans XML, si l'attribut est de type énuméré, ses valeurs possibles sont séparées dans la déclaration par le connecteur de choix `|`.

Les cinq mots clés suivants correspondent aux valeurs par défaut autorisées en SGML mais seules les trois premières sont autorisées par XML.

- `#FIXED` est suivi par une valeur par défaut qui est toujours la même.
- `#REQUIRED` indique que la valeur de l'attribut doit être fournie.
- `#IMPLIED` signifie que si aucune valeur n'est donnée à l'attribut, le système se charge de lui en donner une.
- `#CURRENT` signifie que la valeur de l'attribut est toujours celle donnée pour la première occurrence de la balise de l'élément auquel est rattaché l'attribut.
- `#CONREF` est utilisé avec les attributs qui servent à établir des références croisées à l'intérieur du document.

5.3. Déclarations et appels d'entités

SGML et XML offrent le moyen de définir ou de déclarer certaines parties du documents une seule fois et de les utiliser plusieurs fois. Ceci se fait par le biais de la déclaration et de l'appel des *entités*. Les entités sont utilisées en SGML et en XML de plusieurs façons. Elles peuvent indiquer :

- des notations raccourcies pour des chaînes de caractères dans l'instance (entités générales),
- des caractères spéciaux, des symboles...(entités caractères et entités générales),
- pour inclure des fichiers externes (entités externes),
- des notations raccourcies et des variables dans la DTD (entités paramètres).

Les entités générales et les entités paramètres peuvent être *internes* ou *externes*. Les entités externes peuvent référencer des entités *locales* ou *publiques*. La référence à

4. Pour une description détaillée de la signification et de l'utilisation de ces mots-clés, nous invitons le lecteur à consulter la documentation de référence.

une entité SGML se fait à l'aide de caractères spéciaux : il faudra écrire `&#NomEntité;`, `&NomEntité;` ou `%NomEntité;` selon qu'il s'agit de référencer une *entité caractère*, une *entité générale* ou une *entité paramètre* respectivement.

Les entités générales et les entités paramètres sont référencées en XML de la même manière (que dans SGML). Cependant, deux formes d'appels existent pour les entités caractères. En XML, un appel de caractère (tel qu'un caractère inaccessible par le clavier) fait référence à un caractère particulier du jeu de caractères ISO/IEC 10646. Un appel de caractère se termine toujours par le délimiteur `;`. Il commence par contre par l'un des deux délimiteurs `&#` ou `&#x`. Dans le premier cas, seuls des chiffres se trouvent entre `&#` et `;`. Ces chiffres constituent une représentation décimale de la position du code dans l'ISO/IEC 10646. Dans le second cas, `&#x` et `;` délimitent aussi bien des chiffres que des lettres appartenant à `[a-fA-F]`, et constituent une représentation hexadécimale de la position de code du caractère dans l'ISO/IEC 10646.

De plus, en SGML, si une entité prévue n'est pas encore totalement définie dans la DTD, et que l'on veut analyser cette dernière, le mot clé `#DEFAULT` peut être utilisé pour signaler cela à l'analyseur, comme le montre l'exemple suivant.

```
<!ENTITY E #DEFAULT "Entité non encore définie" >
```

Les entités par défaut ne sont pas autorisées en XML.

5.3.1. Entités caractères

Les documents SGML et XML sont écrits selon un jeu de caractères normalisé, précisé comme *jeu de caractères de base* avant la DTD. Lorsque des symboles ou des caractères supplémentaires sont nécessaires pour le document, ils sont déclarés comme entités caractères par référence à leur jeu de caractères propre (qui correspond en XML à l'ISO/IEC 10646).

Les entités caractères permettent ainsi de définir des notations pour saisir des caractères spéciaux, accents, symboles, etc. Elles sont utilisées pour faire référence à un caractère par son numéro dans un jeu de caractères particulier. Plusieurs ensembles d'entités caractères standardisés ont à cet effet été définis par l'ISO (pour les symboles graphiques, les symboles mathématiques, etc.).

Quand un caractère non SGML est référencé directement par son code, ce dernier est interprété par rapport au jeu de caractères courant, ce qui peut poser des problèmes d'interprétation (de remplacement) lorsque le document contenant est échangé entre deux systèmes n'utilisant pas les mêmes codages. Pour minimiser ces risques, il est conseillé en SGML, d'utiliser les entités générales plutôt que les entités caractères. Ainsi, chaque caractère sera référencé par un nom plutôt que par un code (comme par exemple l'appel `&` plutôt que `&#38;` pour référencer le caractère esperluette (&)).

En XML, les références numériques ne correspondant pas à des caractères de l'ISO/IEC 10646 sont interdites de même que les références aux caractères par des noms sauf pour le jeu des entités générales prédéfinies : amp, lt, gt, apos, quot qui correspondent aux caractères : &, <, >, ' et " respectivement.

5.3.2. Entités générales

Une entité générale est définie par un nom et un contenu. Dans XML, tout nom utilisé dans un appel à une entité doit correspondre à une entité déjà déclarée (sauf pour les entités prédéfinies). Les références aux entités générales doivent être synchrones. Ainsi, aucune entité ne peut être référencée avant d'être déclarée. La déclaration d'une entité générale peut se faire soit dans la DTD, soit au début du document balisé dans le sous-ensemble de déclaration de type de documents *(Document Definition Type Declaration Subset)* à l'intérieur des crochets qui suivent la référence à la DTD par le mot-clé DOCTYPE.

EXEMPLE

```
<!DOCTYPE chapitre SYSTEM "thèse/rapport.dtd" [
<!ENTITY  EPFL     "École Polytechnique Fédérale de Lausanne" >
]>
  ...
<chapitre>
... à l'&EPFL; ...
</chapitre>
  ...
```

Le contenu d'une entité générale peut être :

Un contenu littéral de paramètre *(parameter literal)*. Il est alors formé de données textuelles qui peuvent contenir des balises, des appels d'entités etc., qui seront interprétés au moment où l'entité sera développée.

Une spécification d'entité externe. Le mécanisme d'entités externes offre le moyen d'inclure une partie d'un document (ou d'une DTD) définie dans un autre fichier auquel il faut donner le chemin d'accès. L'entité en question peut être locale au système, le mot-clé SYSTEM est alors utilisé et suivi par une chaîne de caractères contenant des données systèmes (nom de fichier, chemin d'accès, ...). En XML, les identificateurs systèmes sont traités comme des URL. et les entités externes ne peuvent être référencées dans les valeurs d'attributs.
Si la DTD ou l'objet auquel on fait référence possède une déclaration normalisée, le mot-clé PUBLIC est utilisé. Une indication de l'identificateur de dépositaire de texte public est alors requise en plus des données système.
Grâce au mécanisme d'entités externes, le contenu d'un document SGML ou XML peut être réparti sur plusieurs fichiers. Chacun de ces fichiers peut conte-

nir soit des données textuelles, soit des données qui exigent un traitement particulier comme les formules mathématiques, les graphiques, les enregistrements audio ou vidéo numérisés, la musique, etc.
Quand les fichiers externes ne contiennent pas de données SGML, on parle plutôt d'*entités de type de donnée (data entities)*. Pour indiquer le type d'une entité, un mot clé définissant ce type et le nom de la notation[5] doivent être ajoutés après l'identificateur de nom de fichier de l'entité externe. Les références aux entités de type de données sont interdites dans le contenu des documents XML.

Une entité de donnée textuelle. Ce sont des entités générales dont le contenu littéral de paramètre est soit une donnée textuelle `CDATA`, soit une donnée textuelle spécifique `SDATA`, soit une instruction de traitement `PI` *(Processing Instruction)*. Ces entités permettent, par exemple, d'utiliser des symboles particuliers requis par un traitement de texte ou de définir des instructions de traitement pour un système particulier. Les entités de données textuelles (`CDATA`, `SDATA` et `PI`) sont interdites en XML.

Un texte entre crochets. Les textes entre crochets *(bracketed text)* permettent de définir des entités dont le texte de remplacement est du balisage. Selon que l'on utilise le mot clé `STARTTAG`, `ENDTAG`, `MS` ou `MD` dans la déclaration de l'entité, le contenu littéral de paramètre de celle-ci sera encadré par les paires de symboles : < et >, </ et >, <![et]]> ou <! et >. Ces symboles correspondent respectivement aux délimiteurs d'ouverture et de fermeture : de balise de début, de balise de fin, de subsection marquée et de déclaration de balisage. Les entités texte entre crochets sont interdites en XML.

5.3.3. Entités paramètres

Les entités paramètres dans SGML et XML offrent le moyen de construire des objets structurés réutilisables dans les DTD. Les entités paramètres sont utilisées uniquement à l'intérieur des déclarations de balisage, et la plupart d'ente elles se trouvent dans les DTD.

EXEMPLE

Une déclaration d'entité paramètre peut avoir la forme suivante :

```
<!ENTITY % texte "Para|Citation" >
```

L'utilisation de l'entitée déclarée est faite comme suit (en XML il faudra bien entendu supprimer les indicateurs de minimisation de balisage dans la déclaration de l'élément) :

```
<!ELEMENT paragraphe - - (TitreP,(%texte;)+) >
```

5. Une notation est une déclaration dans la DTD qui indique comment les données doivent être traitées.

Utilisées dans une même DTD, ces deux déclarations sont équivalentes à :

```
<!ELEMENT paragraphe - - (TitreP,("Para|Citation")+) >
```

En XML, lorsqu'un appel d'entité paramètre est reconnu dans la DTD et inclus, son texte de remplacement est augmenté d'un espace (`#x20`) au début et à la fin pour s'assurer que le texte de remplacement des entités paramètres contient un nombre entier d'entités lexicales dans la DTD.

5.4. Instructions de traitement

Tout document SGML ou XML est sujet à un traitement automatique par une ou plusieurs applications informatiques. Décrire certains traitements spécifiques directement au niveau de la DTD est possible à travers les *instructions de traitement (Processing instruction)*.

Une déclaration d'instruction de traitement est composée essentiellement des parties suivantes :

- un délimiteur d'ouverture d'instruction de traitement `<?`,
- une instruction de traitement pour un système spécifique,
- un délimiteur de fermeture d'instruction de traitement (`>` en SGML et `?>` en XML).

Une instruction de traitement est destinée à un système bien particulier. Si le document est transmis vers un système non compatible, les instructions de traitement sont ignorées et considérées comme des commentaires. Pour les DTD XML, le système cible (au quel sont destinées les instructions de traitement) doit être explicité dans la déclaration de chacune des instruction de traitement (juste après le délimiteur d'ouverture de déclaration d'instruction de traitement `<?`). L'indication du système cible ne doit cependant jamais correspondre à `[Xx]` `[Mm]` `[Ll]`, ces noms étant réservés au standard XML.

5.5. Commentaires

En génie documentiel tout comme en génie logiciel, les applications[6] ont un cycle de vie et leur maintenabilité est primordiale. Pour une meilleure lisibilité[7] des DTD, SGML et XML offrent un moyen pour les commenter. Un commentaire est déclaré en SGML et en XML sous la forme suivante :

```
<!-- Texte du commentaire -->
```

6. Selon SGML, une application peut désigner une DTD.

7. Une DTD peut être utilisée simplement par un programme informatique, mais elle peut aussi être utilisée directement par un humain.

En XML les commentaires doivent faire l'objet de déclarations autonome, c'est-à-dire qu'ils ne peuvent être inclus dans une autre déclaration (même pas une autre déclaration de commentaire). Ainsi, chaque commentaire est déclaré séparément. Les commentaires vides représentés en SGML par <!> ne sont pas autorisés en XML.

5.6. Déclaration de notation

Le mécanisme d'entités externes permet de rattacher aux documents XML ou SGML des parties de contenu codés dans des formats particuliers et qui doivent être traités par des applications spécifiques. Inclure de tels contenus dans le document lui même plutôt que de les référencer comme entités externes est cependant possible par le mécanisme de déclaration de notation (`NOTATION`).

Ainsi, il devient possible d'inclure, par exemple, dans un document SGML ou XML une formule mathématique codée selon le format de TEX ou LATEX. Le contenu de cette formule sera un contenu d'élément (XML ou SGML) ayant un attribut (par exemple `notation`) dont la valeur correspond à une notation déclarée pour un type d'application particulière.

Un exemple de telles déclarations suit :

```
<!NOTATION TeX   SYSTEM " " >
<!NOTATION LaTeX SYSTEM " " >
<!ELEMENT Equation - - CDATA >
<!ATTLIST Equation notation NOTATION (TeX | LaTeX) "TeX" >
```

5.7. Sections marquées

Dans certains types de documents SGML ou XML, on peut avoir besoin d'inclure des balises sous forme de données plutôt que de balisage (c'est le cas des exemples dans un document traitant du balisage par exemple). De telles portions de contenu ne doivent donc pas être traitées par l'analyseur qui risque de les confondre avec du balisage réel. Un autre type de données particulière sont les portions de contenu destinées à un sous-ensemble particulier (sous certaines conditions) de lecteurs.

La présence de telles données est déclarée par la notion de *sections marquées* (*Marked Sections*). Une déclaration de section marquée contient les parties suivantes :

- un délimiteur de début de section marquée (`<![`) ;
- un mot clé de statut ;
- un délimiteur d'ouverture de sous-ensemble de déclaration ;
- un texte représentant les données ;
- un délimiteur de fermeture de section marquée (`]]`) ;
- un délimiteur de fermeture de déclaration de balisage (>).

En SGML, les valeurs possibles du mot clé de statut sont : `IGNORE`, `INCLUDE`, `TEMP`, `CDATA` et `RCDATA`. Des mots clé multiples peuvent être donnés dans une déclaration de section marquée, dans ce cas, l'ordre de ces mots est le suivant : `IGNORE`, `CDATA`, `RCDATAINCLUDE` (le mot clé `TEMP` n'est pas utilisé puisqu'il désigne généralement une partie temporaire du document qui sera en principe enlevée plus tard).

En XML, le mot clé de statut `TEMP` n'est pas autorisé dans les sections marquées ni le mot clé `RCDATA`. Quant aux mots clé `INCLUDE` et `IGNORE`, il sont interdits dans les instances de documents. Dans chaque déclaration de section marquée, un et seulement un mot clé de statut est obligatoire. Ce mot clé doit être explicite et ne peut donc pas correspondre à un appel d'une entité paramètre.

6. Conclusion

Dans ce papier nous avons présenté une comparaison entre le langage normalisé de balisage généralisé (SGML) et le langage de balisage extensible (XML). Nous avons montré que les documents SGML et XML sont basés sur les notions de DTD et d'instances balisées. XML étant défini comme un sous-ensemble de SGML, nous avons essayé de présenter pour chacune des parties d'un document SGML son correspondant en XML (lorsqu'il existe).

Tout document XML est par construction compatible à la norme SGML. Nous avons vu que moyennant une « personnalisation de SGML » par la mise en œuvre d'une déclaration SGML particulière et quelques contraintes d'utilisation, il devient tout à fait possible de générer des documents XML dans un environnement SGML.

Par ailleurs, la conversion des documents SGML en des documents XML peut sembler difficile. Mais si XML n'est pas syntaxiquement complet par rapport à SGML, il l'est fonctionnellement. En effet, bien que tout document SGML n'est pas un document XML, il y a toujours moyen de faire faire à un système XML ce que peut faire un système SGML en terme de structuration et de modélisation de documents.

XML peut être considéré comme une méthodologie d'utilisation de SGML destinée à en conserver les principales fonctionnalités et à en minimiser les risques d'erreurs et d'ambiguïtés. XML est destié à une classe d'utilisateurs plus large et moins experte que celle des utilisateurs de SGML. XML s'intéresse particulièrement aux internautes conscients de l'importance de la structuration des documents et habituées à un balisage et une navigation facile en HTML. Par l'élimination des fonctionnalités fantaisites et peu utile de SGML, XML rapproche le grand public du pouvoir de structuration des documents par SGML et de tout ce qui peut en résulter pour l'avenir du web.

Bibliographie

[1] Patrick Andries, Samira Cuny, Alain LaBonté, Nicolas Lesbats, François Yergeau, *Langage de balisage extensible (*XML*) 1.0*, Traduction de la recommendation du W3C : REC-xml-19980210 (Reproduite dans ce *Cahier* pages 191–280.). `http://babel.alis.com/web_ml/xml/REC-xml.fr.html`.

[2] Sarra Ben Lagha, *Modélisation et réutilisation de documents structurés*, Thèse de Doctorat (PHD), Ecole Nationale des Sciences de l'Informatique, Tunis, 1998.

[3] James Clark, *Comparison of* SGML *and* XML, W3C Note 15-December-1997, `http://www.w3.org/TR/NOTE-sgml-xml-971215`.

[4] Eric van Herwijnen, SGML *pratique*, Traduction française de Alain Herbuel, Frédéric Orth et Christelle Chaloin, International Thomson Publishing, 1995.

[5] International Organization for Standardization, *Information processing - Text and office systems - Standard Generalized Markup Language (*SGML*)*, ISO 8879, Geneva/New york, 15 october 1986.

[6] World Wide Web Consortium, *Extensible Markup Language (*XML*) 1.0*, W3C Recommendation 10-February-1998, `http://www.w3.org/TR/1998/REC-xml-19980210`.