MASTER EN SCIENCES INFORMATIQUES

Normes et outils SGML pour le traitement de documents électroniques

Jaume, Pascal; Pire, Arnaud

Award date:
1997

Facultés Universitaires Notre-Dame de la Paix
INSTITUT D'INFORMATIQUE

Année académique 1996-1997

Normes et outils SGML pour le traitement de documents électroniques

Mémoire réalisé par Pascal Jaume et Arnaud Pire
en vue de l'obtention du grade de
Maître en Informatique

Stage au Centre de Recherches en Informatique de Nancy (France)
du 1er octobre 1996 au 31 janvier 1997.

Promoteur : Père Jacques Berleur s.j.

Maître de stage : Monsieur Laurent Romary

Abstract

This paper discuss the issue of electronic document manipulations. First of all, we present the SGML and TEI norm which are used at the Centre de Recherches en Informatique (C.R.I.N.) de Nancy *for electronic texts encoding. To manipulate such texts following those norms, one's have to use SGML tools responding to users demands.*

In this paper, we describe functionalities of two SGML tools: a dialog editor and a tree editor. We show that they have the same structure based on three aspects: exportation, manipulation and filtering. The two tools share a set of functionalities. They only contrast by their display.

Ce document aborde la problématique de la manipulation de textes électroniques. Nous présentons tout d'abord les normes SGML et TEI qui sont utilisées au Centre de Recherches en Informatique de Nancy (C.R.I.N.) pour l'encodage de textes électroniques. L'utilisation de textes électroniques selon ces normes réclament des outils de manipulation pour répondre aux besoins des différents utilisateurs.

Nous décrivons les fonctionnalités de deux outils SGML, l'un concernant l'édition de dialogues et l'autre l'édition d'arbres. Les fonctionnalités de ces deux éditeurs sont découpées selon une structure identique, à savoir un niveau filtrage, manipulation et exportation. Les outils sont dotés d'un ensemble commun de fonctionnalités. Ils diffèrent uniquement du point de vue présentation de l'interface.

Keywords : SGML, TEI, *dialogs, corpus, Tree Adjunct Grammar, editors.*

Avant Propos

Avant d'aborder le sujet de notre mémoire, nous tenons à remercier très chaleureusement les différentes personnes qui nous ont aidés à le réaliser :

- Notre promoteur le Père Jacques Berleur s.j., qui a assuré le suivi de nos travaux tout au long de cette année académique ;
- Laurent Romary, notre bien aimé maître de stage qui a su nous motiver durant quatre mois ;
- Patrice Bonhomme, toujours présent pour nous insuffler les subtilités des langages et normes que nous avons utilisés ;
- les deux pauvres utilisateurs : Florence Bruneseaux et Patrice Lopez qui ont bien voulu répondre à nos questions et tester nos applications ;
- Bertrand Gaiffe, manitou des arbres adjoints et grammaires qui a pris le temps de nous expliquer ceux-ci sans se décourager ;
- Jean-Marie Pierrel directeur du C.R.I.N. parce qu'il nous a reçus et introduits dans son équipe scientifique ;
- Chantal Cridlig, la dévouée secrétaire de ce dernier, qui n'a pas de pareille pour régler les nombreux problèmes administratifs auxquels nous avons été confrontés.

Merci à toutes ces personnes, sans qui nous n'aurions pas pu réaliser ce mémoire.

Table des Matières

Table des figures

Tables des tableaux

Introduction

Notre société utilise plus en plus de documents sous une forme électronique. Elle crée des outils permettant d'éditer, de manipuler et d'échanger ces documents. Ces outils respectent différentes normes et langages. Ainsi la norme SGML (*Standard Generalized Markup Language*) a été créée pour permettre de manipuler tout type de documents. Elle permet une structuration sémantique des documents, afin de permettre une manipulation automatique ou du moins semi-automatique de ceux-ci. Cette manipulation a pour objet, non seulement, la présentation des documents, mais surtout leur contenu.

Les domaines d'application sont nombreux : lexiques, dictionnaires, documentations techniques, études linguistiques, Pages Web, etc. L'utilisation de cette norme en tant que telle est rébarbative. Les outils qui la mettent en œuvre doivent permettre une utilisation transparente de la norme. L'utilisateur doit se concentrer sur le cœur même de son travail et pas sur les conventions de codage demandées par la norme.

Nous allons étudier deux outils SGML, afin d'essayer d'en dégager des points communs. Il nous semble en effet possible de trouver un corps de fonctionnalités commun à toutes ces applications, ce qui simplifierait leur développement.

Ces outils ont été développés lors du stage que nous avons effectué au sein de l'équipe Dialogue du Centre de Recherches en Informatique de Nancy (C.R.I.N.) dans le cadre de notre troisième année de maîtrise en informatique aux Facultés Universitaires Notre-Dame de la Paix à Namur (F.U.N.D.P.).

Ces outils ont pour objet l'aide à l'édition de dialogues homme-machine et à l'édition de documents SGML sous forme d'arbres.

Dans un premier temps, nous planterons le contexte général de l'utilisation de la norme SGML et de ses applications en tant que langage de structuration et de manipulation automatique ou semi-automatique de documents électroniques. Nous présenterons également l'environnement dans lequel nous avons travaillé.

Dans un deuxième temps, nous introduirons les deux normes que nous avons utilisées : SGML et TEI (*Text Encoding Initiative*).

Ces descriptions nous permettront de présenter, dans un troisième temps, les contextes particuliers dans lesquels nos deux applications s'inscrivent.

Dans un quatrième temps, nous dégagerons les fonctionnalités nécessaires à la manipulation de documents dans ces deux contextes. Dans ce même chapitre, nous vérifierons que ces deux applications ont des fonctionnalités communes. Enfin, pour clore le chapitre, nous traiterons de la problématique soulevée lors de la mise au point de ces fonctionnalités.

Dans un cinquième temps, nous décrirons les applications que nous avons rédigées, en essayant de développer le plus grand nombre possible de fonctionnalités communes.

1. Contexte

1.1 De l'utilisation de documents sous format électronique et de leur marquage

Si l'objectif de ce mémoire est l'utilisation de SGML au travers d'outils à usages scientifiques, de ses possibilités et de ses limites, il nous paraît nécessaire de situer ce dernier. Nous commencerons donc par parler, dans cette introduction, de la pertinence de manipuler des textes sous un format électronique dans le point " Utilisation et manipulation de textes sous un format électronique ". Nous poursuivrons dans le point " Marquage de textes " en parlant plus spécifiquement des langages de marquage de texte.

1.1.1 Utilisation et manipulation de textes sous un format électronique

A) Utilité

La grande force d'un ordinateur est sa capacité de calcul. La grande force de l'homme est sa capacité d'abstraction. Lorsqu'un homme apprend, il catégorise sa connaissance, il la structure. Lorsqu'un homme lit un texte, l'apprend ou encore y recherche une information, au contraire de l'ordinateur, il n'y voit pas seulement une suite de caractères. Le lecteur y voit des paragraphes, des phrases, des tableaux, des adresses, de l'information, etc.

L'homme, de par sa capacité d'abstraction, associe au texte plus qu'une suite de caractères. Si ce dernier veut se faire aider d'un ordinateur dans ses manipulations de textes, il doit rendre apparent pour l'ordinateur la structure, la sémantique attachée au texte.

Avant tout, la manipulation de documents sous un format électronique rend ceux-ci indépendants des contraintes d'un support solide. Nous n'entendons pas, par-là, qu'un support électronique permet de supprimer la contrainte physique d'un document mais plutôt qu'il le rend virtuel. Il est certain que l'information reste stockée sur un support, mais celui-ci n'est plus le papier, ce qui lui confère quelques qualités non négligeables. Il demande peu d'espace. Il est très facilement reproductible. Il est aisé à transmettre (à condition que l'émetteur et le récepteur possèdent le matériel nécessaire). Il permet des manipulations qui seraient fastidieuses pour un opérateur humain. On peut penser à des

outils tels que des recherches de mots, des copies, des collages, des corrections orthographiques, des analyses statistiques, etc.

1. Quels sont les utilisateurs?

Les utilisateurs de documents électroniques sont de plus en plus nombreux, dans des cadres variés. Cela va du journaliste qui écrit un article et de tous les acteurs qui collaborent à la parution d'un journal, à l'étudiant qui compose son mémoire un mémoire, en passant par le professeur d'université qui veut transmettre le résultat de ses dernières recherches à l'ensemble de ses collègues. Cela concerne également l'expert comptable qui rédige une analyse financière ou la secrétaire qui tape une lettre. Les documents électroniques supportés par les ordinateurs sont devenus un élément indissociable de la vie de tous les jours.

Quand on installe un ordinateur, c'est, dans la majorité des cas, pour y faire de la manipulation de documents. Les utilisateurs ainsi que les documents disponibles sous un format électronique sont de plus en plus nombreux. La masse de documents accessible par un nombre croissant de personnes à travers le monde, grâce à la multiplication des réseaux interconnectés et la présence d'Internet, grandit rapidement. Il en résulte la nécessité de se faire aider par des outils de plus en plus intelligents.

Si les moteurs de recherche sont nombreux et simples d'accès et d'utilisation, ils sont tous très semblables. Ils recherchent des documents sur base de chaînes de caractères, à retrouver grâce à un ensemble de mots clés ou plus largement encore au sein des documents eux-mêmes. Plus il existe de documents accessibles, plus les méthodes de recherche actuelles deviennent insuffisantes. La quantité de documents trouvés ne constitue plus que le résultat d'un premier filtrage. Ce dernier doit être complété par une analyse sémantique des textes trouvés, et ce par l'opérateur humain lui-même, pour en retirer les composants importants.

2. Quels genres de documents ?

Nous allons aborder ici les caractéristiques que l'on peut dégager des différents types de textes qui peuvent être utilisés (écrit, parlé, autres).

a. Textes écrits

Le codage des textes écrits permet la constitution de librairies électroniques contenant des versions électroniques de romans, de revues, de documentations techniques, etc.

Une autre application intéressante est l'alignement multilingue. Ce principe consiste à mettre en parallèle des portions assez fines de deux textes, l'une étant la traduction de l'autre. On peut, dès lors, par exemple, comparer phrase par phrase la traduction d'un texte.

b. Textes parlés[1]

Un *spoken text* est une représentation écrite ou électronique d'un morceau de discours (langage parlé) défini comme une unité en vue de divers traitements.

Exemples:

- interactions face à face ;
- conversations téléphoniques ;
- interviews ;
- débats ;
- commentaires ;
- discours.

On peut encore ajouter à cette liste des formes spéciales de *spoken texts* telles que les *scripted speeches* (émissions télévisées, lectures à haute voix), et les dictées.

On peut dire que réussir à coder l'interaction face à face revient à pouvoir coder toutes les autres formes de *speeches*. En effet, l'interaction face à face est spontanée, interactive et dépendante de la situation.

Différents types de transcriptions

Les *speeches* sont transcrits par de nombreuses personnes, qui adoptent une transcription adaptée à leurs besoins. Ces transcriptions peuvent être utilisées dans différents cadres. En effet, elles peuvent servir à l'étude de la langue elle-même ou, ce qui plus courant, à la manipulation du contenu du document. Nous les dénommerons respectivement transcription linguistique et transcription non linguistique.

Transcriptions linguistiques

Les transcriptions linguistiques peuvent être de type lexicographique, phonétique, etc. Elles peuvent également concerner la transmission de la parole.

[1] [JOHAN, 95].

La transcription varie en fonction du type d'utilisateur. Dans les applications non linguistiques destinées à une large diffusion, la transcription diffère assez peu d'un texte ordinaire.

Dans les applications linguistiques par contre, l'attention est plutôt portée sur les caractéristiques spécifiques du *speech* : à savoir, les caractéristiques prosodiques et paralinguistiques, le recouvrement des locuteurs, les pauses, les hésitations, les répétitions et les interruptions.

Transcriptions non linguistiques

Les personnes utilisant des transcriptions non linguistiques sont notamment les journalistes, les psychologues, les spécialistes en sciences humaines, les transcripteurs d'interrogatoires, de procédures judiciaires ou encore les transcripteurs de comptes rendus parlementaires, de conférences, etc.

Problèmes liés au codage de *speeches*

Il y a beaucoup de problèmes spécifiques liés à la transcription du *speech*. Le *speech* varie selon diverses dimensions qui n'ont pas de contrepartie dans le texte écrit (tempo, bruit,...). Ainsi, la qualité d'un speech enregistré peut ne pas être parfaite et affecter de ce fait l'exactitude de la transcription. Comment représenter ce phénomène?

Qu'en est-il des gestes associés au *speech*?

De plus, le transcripteur doit prendre en compte le contexte dans lequel le *speech* a eu lieu et ainsi fournir un certain nombre d'informations.

c. Autres documents

Il existe de nombreux autres documents qui ne sont pas seulement des documents écrits ou parlés. On peut, en effet, insérer dans un document des photos, dessins, films, liens hypertextuels, etc. On cite ainsi :

- documents multimédias ;
- pages Web ;
- grilles horaires ;
- recueils de peintures, de photos ;
- partitions musicales ;
- formules mathématiques et chimiques ;
- etc.

Un texte uniquement écrit peut, par sa structure, donner naissance à des contraintes autres que celles engendrées par un texte habituel. C'est le cas, notamment, du dictionnaire.

La constitution d'un dictionnaire électronique est un travail très complexe. Les dictionnaires figurent en effet parmi les textes les plus difficiles à traiter. Chaque entrée d'un dictionnaire est un objet fortement structuré dans lequel de nombreux mécanismes d'abréviations permettent une présentation condensée des informations. La structure des entrées des dictionnaires varie fortement d'un dictionnaire à l'autre et ainsi qu'au sein d'un même dictionnaire. On peut trouver tout type d'information à n'importe quelle position. Il faut donc un principe de codage permettant d'appréhender les variations liées à la structure du dictionnaire utilisé.

Un autre problème de codage provient du fait que les dictionnaires sont à la fois des textes et des bases de données. En effet, les utilisateurs ne lisent pas le dictionnaire de A jusqu'à Z, mais accèdent à des entrées à partir d'une clé. Ce genre d'accès est typique des bases de données.

De plus, il est clair qu'un dictionnaire électronique se doit d'offrir d'autres types d'accès que celui de la clé principale . Ainsi, l'utilisateur doit pouvoir accéder à tous les mots dont la définition contient un mot donné. Il doit également pouvoir obtenir la liste de tous les mots remplissant certains critères (exemple : rechercher tous les noms relevant du domaine informatique avant 1960) [GUT, 96].

B) Les formats disponibles

Les documents électroniques abondent et nombreux sont les moyens de les représenter. En voici une présentation succincte.

1. Textes bruts

Les formats initiaux pour les documents électroniques reposaient et reposent toujours sur des normes telles que l'ASCII[2]. Ce standard, encore énormément utilisé sert à coder des caractères sous forme binaire. Au départ ces derniers étaient représentés sur sept octets, puis sur huit. Cette représentation binaire permet aux ordinateurs de manipuler les caractères. En conséquence, on pourra caractériser cette manipulation "d'automatique ".

[2] *American Standard Code for Information Interchange*

Si ces formats permettent la manipulation de textes, leur sauvegarde sur support magnétique, leur transfert électronique et leurs manipulation en mémoire, ils ne permettent cependant pas la manipulation des unités grammaticales, des mots, des phrases, des paragraphes ou encore des documents à proprement parler. En effet, la caractéristique sémantique apportée à ces notions ne transparaît pas au travers de ces normes.

D'aucuns pourraient suggérer la possibilité de recomposer automatiquement ces diverses notions[3]. En effet, un document commence par un caractère précis, et se termine par un autre. Les phrases commencent par une majuscule et se terminent par un point et les paragraphes sont suivis d'un saut de ligne. Quant aux mots, ils sont suivis d'un espace.

C'est sans compter sur la diversité et la complexité des langages humains. S'il est vrai que l'on peut se contraindre à respecter ces quelques règles simples lors de l'écriture ou la retranscription de texte, cela devient beaucoup moins vrai lorsque l'auteur prend des libertés avec la ponctuation, lorsqu'il utilise des acronymes (comment distinguer les points entre les lettres et les points de fin de phrases) ou qu'il veut transformer un document sous forme verbale en un document écrit. Comment tenir compte des hésitations qu'un orateur a émises lors d'un discours ? Quand la phrase se termine-t-elle ?

On comprend ainsi aisément qu'il est nécessaire de définir un codage d'ordre supérieur à ceux orientés caractères. Notons, cependant, que ce codage d'ordre supérieur devra utiliser ceux orientés caractères.

2. Les différents formats associés aux éditeurs de textes

Les formats orientés caractères n'ont pas seulement été créés pour la beauté du geste, mais pour répondre aux exigences de la manipulation de textes. Aux symboles contenus dans les différents alphabets de la planète, furent adjoints des symboles dits « de contrôle » permettant une première structuration des textes, leurs affichages et leurs impressions : points, virgules, fin de textes, sauts de lignes, sauts de pages, etc. On leur adjoint également d'autres symboles permettant d'exprimer des faces souriantes, des flèches et autres "@™*f*„‡¥‰$%&®¿ ".

Ces ajouts sont bien insuffisants à la bonne mise en forme des textes. Chaque concepteur d'éditeur de textes créera donc des suites de caractères dits d'échappement, qui permettent la mise en gras, en souligné, en italique, en exposant, en barré, etc. Si ASCII est

[3] Il s'agit évidemment d'un sujet de recherche actuel. Citons notamment la thèse de doctorat de Patrice Bonhomme (C.R.I.N.) en cours de rédaction.

devenu une norme, il n'en est pas de même pour les formats de textes qui ont été manipulés par un éditeur. Il existe, également, différentes séquences d'échappement propres à chaque imprimante qui permettent l'impression effective de ces documents mis en formes.

Quoi qu'on en dise, même s'ils sont différents les uns des autres, ces formats existent. Malheureusement, ils ne permettent que la mise en forme - en vue de l'impression - des documents. Ils n'aident en rien à la manipulation sémantique à proprement parler des documents. En effet, rien ne nous permet de déduire (et encore moins à la machine), qu'une suite de chiffres présente dans un document est en fait un numéro de téléphone qui pouvent être composé pour joindre l'auteur. Rien ne permet, non plus, de déduire que la suite de mots que l'on vient de traiter est en fait la référence à un autre document dont la lecture aiderait à la compréhension du texte. On comprendra, donc, la nécessité de structurer plus encore le document pour pouvoir retirer le maximum de son contenu sémantique, de façon automatique ou semi-automatique.

3. Vers le marquage des textes

L'idée est donc née d'insérer dans le texte un marquage sémantique, qui tenterait de couvrir tous les aspects de la production humaine de documents. De plus, il serait opportun de se séparer des contraintes apportées par les différentes normes citées plus haut, de ne plus se préoccuper du type d'ASCII, qu'il soit latin I ou II, anglais ou autre. Ce format devrait indiquer, non seulement, la présence d'un numéro de téléphone, mais aussi celle d'une formule mathématique ou d'une référence à un texte, à un autre chapitre. Celui-ci permettrait d'obtenir de façon claire des listes d'objets, qui pourraient être manipulées automatiquement. Un objectif supplémentaire à atteindre serait de pouvoir vérifier la cohérence des éléments d'une liste par rapport à une collection de critères.

Le marquage de textes pourrait non seulement prendre en compte les différentes façons de présenter un texte mais surtout permettre de le structurer sémantiquement. Ce format apporterait un sens aux suites de mots contenues dans un document. Il permettrait une manipulation différente d'un document en fonction de son contenu : liste de numéros de téléphones, un poème, un essai, etc.

C'est dans ce but que la notion de marquage de textes a été développée. Celle-ci sera abondamment illustrée et expliquée au chapitre 1.1.2.

4. Autres moyens que le marquage

Citons deux autres normes permettant de structurer un document : O.D.A. (*Office Document Architecture*) OpenDoc (*Open Document Specification*).

O.D.A. définit une architecture pour les documents du business en termes de sa structure logique et de sa présentation. Il existe peu d'applications du standard ODA. S'il convient à des documents tels que des livres ou des correspondances professionnelles qui utilisent une présentation rectangulaire, il ne convient pas pour des documents tels que des catalogues, des brochures, des journaux, des magasines dont la présentation est souvent irrégulière et diagonale [ODA, 97].

OpenDoc permet de spécifier le contenu et la structure de documents. Il supporte des fonctions telles que la superposition de textes et d'images, des textes d'aspects irréguliers et d'objets textes insérés.

1.1.2 Marquage de textes

Le marquage rend explicite pour l'ordinateur des choses qui sont implicites pour le lecteur humain. Il rend explicites les informations contenues dans le texte, que ce soit pour l'affichage, l'impression ou la sémantique même de l'information.

Les textes peuvent être très complexes, ils peuvent contenir des annotations, des mots ou des phrases dans une langue étrangère, des jeux de caractères différents et spéciaux. De plus, les textes sont structurés différemment selon qu'il s'agit de discours, de poèmes, de dialogues ou encore de documentations techniques. Il faut faire en sorte que tous ces éléments transparaissent dans le document électronique.

Plusieurs manières de coder les textes ont été développées. Cependant, on s'est vite rendu compte que ce développement anarchique ne menait à rien. En effet, certains développements se concentraient sur des types de textes particuliers, d'autres se concentraient sur les aspects typographiques. Ce type d'approche (typographique) provoque souvent des ambiguïtés, car un mot en italique peut être un titre, un mot provenant d'une langue étrangère ou encore un mot sur lequel on veut insister.

C'est ainsi que fut décidée la création d'un "*encoding scheme*" flexible, qui puisse prendre en compte tous les éléments spécifiques liés à un texte. Le même texte électronique peut être utilisé pour différents usages. Cela nécessite un "*encoding scheme*"

complètement indépendant du software et du hardware, qui puisse être transmis à travers les réseaux.

L'expérience montre qu'une des principales difficultés posées par la manipulation de textes sous une forme électronique est le choix du codage ou du format utilisé. En effet, de ce choix dépend, en grande partie, le succès d'une étude, d'une recherche linguistique, du stockage ou de l'échange de toute ressource textuelle. Nous noterons l'intérêt du marquage en tant que norme pour l'échange des données et la définition d'outils standardisés.

Echange des données - Au moment où les autoroutes de l'information sont en plein essor, il est important de ne pas oublier la diffusion ou l'échange de ces ressources. Mis à part le *Web*, le support peut être aussi varié qu'une disquette, une bande magnétique ou un CD-ROM. L'échange doit alors être effectué dans les meilleures conditions possibles afin de ne pas perdre d'informations ou de ne pas détériorer la structure de ces données. Le choix du codage devra alors se porter vers un format portable d'un support à un autre, d'une architecture à une autre.

Définition d'outils - Quelle que soit la forme sous laquelle se présente le texte, celui-ci ne sera utilisable que s'il existe des outils pour le manipuler. Un certain niveau de normalisation permettra la mise à disposition de tout utilisateur d'un ensemble d'outils standardisés (commandes spécifiques), et évitera la duplication d'outils ayant la même fonction mais sous des formats différents.

Le marquage doit préserver les différents éléments de la structure du texte, permettre la recherche de certaines caractéristiques, aider à naviguer et à utiliser le texte. Comment rechercher un chapitre particulier dans un roman s'il n'y a pas un marquage indiquant le commencement et la fin des chapitres? Comment rechercher un mot dans une pièce de théâtre dont on connaît la scène, si les scènes ne font pas l'objet d'un marquage.

Prenons un exemple : lors de la conversion d'un fichier de texte dans un autre format, il arrive souvent que des caractéristiques soient perdues telles que l'alignement, les caractéristiques attachées à un mot (italique, gras, souligné, etc.), la police d'écriture, les lettres accentuées, etc. Il en résulte une perte de sémantique.

Le marquage est constitué d'un ensemble de lettres et de caractères adjoints au texte même, qui voyagent avec le texte quand il passe d'un système à un autre [BONHOM, 96].

A) Que veut-on marquer ?

Comme nous l'avons dit plus haut, le marquage va nous permettre d'apporter une certaine sémantique à notre document. Il va de soi que le linguiste ne va pas voir un document (un livre par exemple) de la même façon que l'auteur ou qu'un lecteur ordinaire. L'auteur voudra y voir, en plus du texte que le lecteur lira, des annotations, des références, etc. Le linguiste, pour ce qui le concerne, voudra y voir apparaître des informations propres à sa profession : construction des phrases, des mots, références linguistiques, etc. Le lecteur ne voudra voir qu'un texte bien mis en forme. Un internaute voudra pouvoir *surfer* et suivre les références contenues dans le texte au travers du *Web*.

Chaque lecteur veut voir ce texte sous un aspect particulier qui n'intéresse aucun autre type de lecteurs. Le marquage devra donc être dédicacé à son lecteur. Il est donc impossible de prévoir toutes les composantes de ces facettes, pour toutes les catégories de lecteurs imaginables. Il faut donc prévoir une méthode qui permette de concilier ces visions. Chaque catégorie de lecteurs devra pouvoir créer son marquage propre, sa propre norme.

B) Les normes existantes

De nombreuses normes de marquages existent, le Tableau 1-1 en présente quelques unes. Notons que SGML n'y figure pas. En effet, SGML n'est pas utilisé en tant que tel pour marquer des texte. Ce sont plutôt des langages dérivés de celui-ci qui sont utilisés. Parmi les langages repris dans le tableau HTML, HYTIME et TEI font partie de ces langages dérivés. Nous expliquerons, au chapitre 2.1, en quoi consiste précisément la norme SGML.

Langage	Description
HTML	Un langage de marquage qui permet la mise en forme de document sur Internet.
HYTIME	Un langage de marquage scientifique et technique très puissant [HYTIM, 97].
ROFF et dérivés	Permettent la structuration de texte en vue de leur utilisation pour les manuels Unix (man).
RTF	Rich Text Format - représente toute la mise en forme d'un document. Il convertit la mise en forme en instructions pouvant être lues et interprétées par de nombreuses applications.
TEI	Un langage de marquage de manipulation de texte qui sera présenté au chapitre La *Text Encoding Initiative*.
TEX/LATEX	Au départ, conçu pour faire de l'édition scientifique (imbattable dans le domaine des formules).
XML	Extensible Markup Language. Une version édulcorée de SGML qui est en cours de définition au sein du consortium WWW pour faciliter les échanges de documents structurés sur le Web.

Tableau 1-1 - Autres normes de marquage

C) La notion de corpus[4]

Le terme corpus désigne habituellement une collection de données linguistiques, comprenant des écrits, des documents parlés, ou les deux, dans une ou plusieurs langues. Dans certains cas, ce terme est plus restrictif et s'applique aux collections correspondant à divers critères linguistiques. Un corpus peut contenir de multiples formes de textes : de la prose, des journaux, de la poésie, des pièces de théâtre, ainsi que des listes de mots, des dictionnaires, etc.

Tous ces documents sont encodés dans le but de servir de ressource en ingénierie de la langue, incluant toutes les zones de la manipulation automatique du langage naturel, de la traduction automatique, de la lexicographie, etc. L'intérêt est de rassembler quantitativement et qualitativement de exemples réels de la langue qui peuvent servir à la construction de lexiques, de grammaires, de lexiques multilingues, etc.

Les informations quantitatives permettant des analyses statistiques peuvent être tantôt utilisées pour guider des *Parsers* basés sur des préférences, tantôt pour assister à la construction de lexique ou encore pour déterminer des correspondances de traductions.

[4] [CES,96].

Ces corpus peuvent également être utilisés à des fins d'analyses stylistique, socio-linguistique ou historique, ainsi qu'à la recherche d'informations.

1.1.3 Conclusion

Nous avons, dans cette partie " De l'utilisation de documents sous format électronique " tenté d'expliquer dans quel environnement se trouve le marquage de texte et plus particulièrement le langage SGML.

Il nous paraissait important d'expliquer en quoi consistait un langage de marquage, par qui et à quels desseins SGML était utilisé, avant d'en donner une explication. La compréhension de ce contexte permettra au lecteur de mieux comprendre la finalité d'une telle norme, ainsi que les principes qui sont abordés lors de la création d'une application qui l'utilise.

1.2 Environnement scientifique

Cette introduction se doit enfin d'évoquer notre environnement de travail. En effet, nous tenons, dans ce document, à aborder la problématique de SGML et de ses outils dans le cadre d'un environnement de travail scientifique. Nous présenterons donc successivement le centre de recherche et l'équipe dans lesquels nous avons travaillé ainsi que le rôle que l'on nous a demandé d'y jouer.

1.2.1 Le C.R.I.N.[5]

Le Centre de Recherches en Informatique de Nancy, dirigé par Jean-Marie Pierrel, est associé au C.N.R.S. et à l'U.R.A.[6] 262. Il s'agit d'un laboratoire commun au trois universités de Nancy.

La politique de recherche du C.R.I.N. repose sur six axes.

A) Communication homme-machine et intelligence artificielle

> Cet axe concerne la définition de nouveaux modèles permettant d'accroître les capacités de raisonnement et de communication des ordinateurs avec l'homme, et le test de ces modèles dans des environnements réalistes :

[5] [CRIN, 96]

[6] Unité de Recherche Appliquée.

- Dialogue homme-machine,
- Recherche interactive dans les systèmes d'information multimédia, modélisation d'informations complexes,

B) Image, modélisation et simulation

Etude de problèmes d'analyse et de synthèse d'images posés par les applications de réalité virtuelle et la modélisation de surfaces naturelles complexes ouvrant la voie à une nouvelle forme de CAO.

C) Perception et raisonnement

Etude sur l'intelligence artificielle dans le cadre général de l'interaction homme-machine. Elle s'intéresse aux processus cognitifs de perception et de raisonnement, ainsi qu'à la conception et à la réalisation d'architectures de systèmes à bases de connaissances.

D) Logiques, preuves, résolutions de contraintes et algorithmes

Résolution de problèmes concrets issus de l'ingénierie industrielle en utilisant des modèles géométriques ou discrets, déterministes ou stochastiques. Recherches sur la preuve de systèmes informatiques matériels et logiciels. Conception et réalisation d'outils permettant d'intégrer le développement de programmes et les preuves de leurs propriétés.

E) Réseaux, parallélisme et distribution

Etude de méthodes et de formalismes pour la spécification, le raffinement et le développement de programmes ou de systèmes. Application de ces techniques à des domaines comme le parallélisme, les systèmes distribués, les télécommunications, les systèmes sécuritaires.

F) Construction de logiciels

Modélisation de procédés de construction de spécifications et de programmes afin de développer des programmes sûrs.

Les travaux entrepris au C.R.I.N., notamment par l'équipe DIALOGUE, relatifs au marquage de textes et plus particulièrement à SGML se situent dans le premier axe de recherche.

1.2.2 L'équipe DIALOGUE[7]

A) Introduction générale

L'objectif de l'équipe dialogue est de progresser dans la définition et la mise en œuvre de systèmes de communication homme-machine robustes et fiables, à forte composante langagière. En effet, l'introduction du langage naturel et plus particulièrement de la parole dans une interface homme-machine nécessite des systèmes à la fois robustes du point de vue de la reconnaissance et de la compréhension du langage, et bien intégrés à l'application mise en œuvre.

B) Problématique

Concevoir des interfaces agréables et surtout adaptées à leurs conditions d'emploi pour des tâches particulières est un objectif bien difficile à réaliser. Les recherches conduites au sein de l'équipe DIALOGUE rassemblent un ensemble d'études visant à intégrer la langue dans des systèmes de dialogues finalisés pour une tâche donnée. Dès lors, on y retrouvera des actions de recherche autour des niveaux classiques de l'étude de la langue : reconnaissance de la parole jusqu'à un premier niveau linguistique (phonème[8] ou mot selon les cas), sémantique depuis le lexique[9] jusqu'à l'énoncé et enfin pragmatique[10] avec en particulier des études sur la référence aux objets et aux actions.

L'objectif de l'équipe nécessite l'étude de l'usage de la langue en situation, ce qui pose, en particulier, le problème de l'interaction entre langue et gestes, et celui de l'interaction entre structures langagières et la représentation d'une application utilisant le dialogue comme moyen d'interaction. Il y a donc deux perceptions de la langue : l'une est la langue comme moyen d'interaction et l'autre est la langue comme moyen d'information. L'équipe travaille sur ces deux perceptions. Il faut néanmoins préciser que ces perceptions, et donc les travaux effectués sur chacune d'elle, sont complémentaires.

---

[7] [DIAL, 97]

[8] Un phonème est un segment phonique qui : (a) a une fonction distinctive, (b) est impossible a décomposer en une succession de segments dont chacun possède une telle fonction, (c) n'est défini que par les caractères qui, en lui, on valeur distinctive, dits pertinents [DUCROT, 72].

[9] Ensemble des mots ayant une valeur de dénomination et formant la langue d'une communauté, d'une activité humaine, d'un individu.

[10] La pragmatique décrit l'usage que peuvent faire des formules, des interlocuteurs visant à agir les un sur les autres [DUCROT, 72].

1. La perception de la langue comme moyen d'interaction amène à considérer des applications interactives (informatiques ou autres) engendrant des situations de dialogue, dans lesquelles un usager souhaite converser avec le système par le biais de la langue. L'interaction peut ne pas être purement langagière et associée à des gestes. La réalisation d'une telle application suppose une analyse des énoncés en rapport avec une certaine capacité ou complexité expressive du langage d'un utilisateur dans une telle situation. L'interprétation de cet énoncé doit être faite dans le contexte de l'application qu'on est amené à représenter.

Dans la perspective d'une communication langagière, des études sur les liens entre la langue et l'environnement dans lequel se déroule le dialogue sont menées. Cette action nécessite en particulier des recherches pluridisciplinaires importantes en pragmatique, aux confins de l'informatique et de la linguistique, qui rendent indispensable la présence au sein de l'équipe d'un linguiste, spécialiste de ce domaine.

La langue, mise dans la perspective de l'interaction, n'a de sens que vis-à-vis d'êtres humains utilisateurs du système. L'usage effectif que font ces utilisateurs des systèmes réalisés ou projetés motive les actions de recherche liées à l'ergonomie du dialogue à forte composante langagière.

En particulier, le traitement complet du langage naturel n'étant pas actuellement possible, on est amené à définir des sous-ensembles restreints de celui-ci. La compréhension de l'interaction entre l'utilisateur et le système, au travers de ces sous-ensembles, s'avère être un point important pour le traitement complet du langage naturel.

Ces recherches, de par leurs objectifs, conduisent à aborder trois aspects complémentaires :

- le traitement de la référence, fondamental pour l'interprétation de dialogues utilisés comme moyen d'interaction ;
- les mécanismes de dialogue indispensables pour apporter une aide véritable aux usagers ;
- Les architectures informatiques à mettre en œuvre pour l'implantation de tels systèmes de dialogue.

2. La perception de la langue comme support de l'information apporte au point précédent, d'une part la capacité de réaliser la chaîne d'analyse des énoncés par des travaux sur la reconnaissance de la parole et la prise en compte de la prosodie[11] et, d'autre part, la possibilité d'articuler les niveaux linguistiques et conceptuels dans des univers techniques et applicatifs. Dans la perspective de l'utilisation de systèmes de dialogue, en particulier multi-modaux, l'usage effectif de la langue repose essentiellement sur l'oral. L'équipe DIALOGUE mène sur ce point trois actions de recherche portant respectivement sur les liens entre modèles articulatoires et phonétiques, sur la prosodie et sur les méthodes de reconnaissance de parole. Ces trois points relèvent de la langue comme support d'information dans la mesure où ces recherches restent indépendantes d'une situation de dialogue particulière. Par ailleurs, le domaine de la documentation technique produit des masses de documents en langue. Même s'il semble clair que la consultation de ces documents suppose des systèmes de dialogue adaptés, une première étape vise à extraire l'information de ces documents, et en particulier leur terminologie.

L'équipe mène une action de recherche centrée sur la mise en forme de corpus de textes et/ou de transcriptions de dialogues dans une double perspective d'étude de la langue à des fins de dialogue et d'extraction d'information.

C) Principaux thèmes de recherche

Les actions de recherche qui suivent peuvent se concevoir à la fois comme relevant de l'étude de la langue comme support d'information, mais aussi dans la perspective de fournir des modèles et des outils visant à utiliser la langue pour le dialogue.

1. Reconnaissance de la parole

Le travail est mené dans une optique de reconnaissance et de compréhension, de la parole et du langage naturel. Il a pour but de prendre en compte de manière réaliste les mécanismes de production de la parole. L'objectif de l'équipe est le passage du traitement du signal de la parole à la pragmatique. Les connaissances utilisées sont le décodage acoustico-phonétiques multi-locuteurs de la parole, la reconnaissance de mots, la compréhension de phrases, la prosodie, les informations structurelles telles que la syntaxe[12], la sémantique[13], la modélisation et l'interprétation de dialogues.

[11] La prosodie est l'étude des sons du langage dont les composantes sont le timbre, la hauteur, l'intensité et la durée [DUCROT, 72].

[12] La syntaxe consiste à déterminer les règles permettant, en combinant les symboles élémentaires, de construire des phrases ou formules, correctes [DUCROT, 72].

Le travail sur la reconnaissance de la parole est scindé en deux axes :

- Le premier axe consiste à modéliser les aspects temporels de la parole dans un système neuromimétique[14], de déclencher un perceptron multicouches[15] à l'endroit où son utilisation sera la plus discriminante possible, et enfin d'utiliser des automates sans topologie fixée à priori qui permettent d'exploiter au mieux le corpus d'apprentissage.
- Le second axe consiste à utiliser les connaissances phonétiques et articulatoires dans le cadre d'un système de décodage acoustico-phonétique. Cette axe porte également sur l'étude de la prosodie car elle peut contribuer à la segmentation du signal de parole.

Dans le cadre du premier axe, et plus précisément dans la reconnaissance de mots isolés, le CRIN a développé une méthode à base d'automates dans laquelle chaque mot du vocabulaire est représenté par un automate, chaque chemin de l'automate décrivant une variation d'élocution de ce mot. Contrairement aux autres méthodes existantes, l'approche développée permet facilement un apprentissage incrémental. En effet, lorsqu'un mot est mal reconnu, il suffit d'ajouter un chemin représentant ce mot dans l'automate.

Un des buts poursuivis dans ce deuxième axe est la construction d'un système de DAP[16] dont le principe est d'expliquer de manière cohérente les observations acoustiques en fonction des connaissances des phénomènes de production de la parole. Un autre point important est l'étude des variations temporelles des paramètres prosodiques (rythme, intonation, intensité) au cours de la production d'un énoncé qui contribuent, dans une large mesure, à l'identification de la structure syntaxique et discursive de cet énoncé par l'auditeur. L'objectif est dans un premier temps, de construire des algorithmes robustes de détection et d'interprétation des informations linguistiques et énonciatives véhiculées par la prosodie en parole continue et d'intégrer ensuite ces algorithmes dans un système de reconnaissance de la parole continue.

[13] La sémantique vise à donner le moyen d'interpréter les formules syntaxiques, de les mettre en correspondance avec d'autres choses, cet "autre chose" pouvant être la réalité, ou bien d'autre formules (de ce même langage ou d'un autre) [DUCROT, 72].

[14] qui imite l'action des neurones.

[15] Type particulier (le plus simple) de réseau neuronal.

[16] le Décodage Acoustico-Phonétique permet de passer du signal de parole à une représentation sous la forme d'une suite de phonème.

2. Terminologie et textes techniques

Cette activité répond à un besoin croissant, tant au niveau industriel que scientifique, de maîtrise de volumes très importants de documentations scientifiques et techniques. L'objectif est de développer des méthodologies et des outils informatiques et linguistiques pour le traitement de ces textes. La notion de traitement est prise dans un sens relativement vaste puisqu'il peut s'agir d'apporter de nouvelles données pour des problématiques comme :

- la recherche d'information ;
- la synthèse de textes scientifiques ;
- l'aide à la rédaction ;
- l'analyse automatique de textes techniques en vue de contrôler leur qualité.

3. Traitement de la référence dans la langue

L'équipe a pour but de mettre en évidence les mécanismes liés à la référence en situation de dialogues et d'énoncés en langage naturel. L'approche repose d'une part sur une analyse fine des indices linguistiques disponibles et d'autre part sur la prise en compte du caractère fortement localisé (dans le temps et l'espace en particulier) des interprétations auxquelles ces indices conduisent.

Les travaux menés jusqu'ici ont conduit à une proposition de modèle pour l'interprétation des expressions référentielles dans un dialogue homme-machine : celle-ci repose sur l'hypothèse que ces expressions ne visent pas directement des référents dans un historique de dialogue ou un environnement perceptuel partagé. Au contraire, elles doivent s'interpréter comme des instructions de sélection à l'intérieur de contextes locaux.

Dans tout travail sur la référence, la notion d'espace de référence est capitale. L'équipe travaille à l'intégration, sur la base de travaux précédents concernant la référence, de l'ensemble des mécanismes qui permettent d'accéder à un espace visuel (graphique) à l'aide d'expressions multimodales à forte composante langagière. Une bonne définition de l'espace de référence passe notamment par:

- la modélisation du geste de désignation dans le cadre d'énoncés où celui-ci intervient comme complément à l'expression langagière;
- la définition de modes de représentation des objets pour permettre d'interpréter au mieux les gestes leurs correspondants;
- le filtrage des informations contextuelles utiles à l'interprétation d'une expression donnée.

Même si, comme il est dit plus haut, les travaux sur les expressions référentielles portent sur des contextes locaux, on ne peut pas analyser ces dernières en faisant l'impasse sur la propriété qui leur est commune. A savoir leur capacité à désigner un objet dans le monde, et donc hors du discours où elles apparaissent. Cette hypothèse découle d'un certain nombre de thèmes communs :

- importance d'un traitement des expressions référentielles qui ne soit pas limité à la linguistique strictement entendue (syntaxe et sémantique) ;
- nécessité de l'intégration de données non linguistiques, qu'elles relèvent de la connaissance sur le monde ou de la situation de communication.

Au-delà des références liées au texte proprement dit ou aux objets, la référence aux actions soulève également des problèmes. Classiquement, dans le domaine du dialogue de commande homme-machine, les énoncés sont analysés en termes de structures prédicatives, à charge pour la description lexicale d'établir le lien entre les prédicats et les fonctions de l'application à exécuter. Des difficultés apparaissent donc dès qu'un énoncé se réfère à une séquence de fonctions et non plus à une fonction isolée. Ce type de situation apparaît de façon naturelle, en particulier lorsqu'il s'agit de créer des objets. Ainsi, un système classique accepterait des énoncés tels que « Déplace le carré rouge » ou encore « Crée un carré », mais refuserait « Crée un carré rouge » pour autant que l'application ne fournisse pas une fonction permettant de créer un carré d'une couleur donnée. Si le concepteur veut permettre ce type d'énoncé, il doit créer une nouvelle fonction. Il a été prouvé que le nombre d'opérateurs à ajouter est une fonction exponentielle du nombre de propriétés utilisables pour la référence. Ce type de démarche risque donc bien souvent de conduire à des applications incomplètes et incohérentes. La proposition faite consiste à modéliser les énoncés comme référant à un état final, à charge pour un planificateur de calculer la séquence d'actions à mettre en œuvre pour l'atteindre.

4. Dialogue et applications

Offrir à l'usager la possibilité d'interagir oralement avec un logiciel d'application soulève des problèmes spécifiques non résolus actuellement. Les travaux en cours portent sur les thèmes suivants :

- l'analyse ergonomique de l'usage de la parole et du geste en situation d'interaction personne-machine, et la modélisation du comportement et des stratégies d'expression multimodale des futurs usagers ;
- la conception d'architectures logicielles et d'outils génériques destinés à faciliter le développement d'interfaces utilisateur robustes et conviviales, qui intègrent la parole parmi les modalités d'expression offertes à l'usager.

Notre rôle au sein de l'équipe dialogue fut de développer deux applications. La première avait pour objectif de mettre au point un logiciel permettant l'édition d'arbre SGML, cet éditeur devant à terme apporter à Patrice Lopez une aide dans son travail de recherche sur les grammaires d'arbres adjoints.

La seconde application devait être un outil permettant la manipulation de corpus de dialogues, sur lesquels travaille Florence Bruneseaux, afin de mettre en évidence les mécanismes de référence au travers de ces dialogues. Nous avions à notre disposition la librairie de fonctions *Dilib* qui contient un ensemble de fonctionnalités permettant la manipulation de documents SGML. Ces développements devaient se faire en C pour la partie manipulation de données et en Tcl/Tk pour la partie interface.

2. Les normes

Ce chapitre présente les différentes normes et langages que nous avons utilisés. Leur présentation donnera les notions nécessaires à la compréhension des deux applications sur lesquelles nous avons travaillé et que nous développerons dans le troisième chapitre.

Dans une premier temps, nous approcherons la norme SGML dont nous avons déjà décrit la philosophie générale dans le premier chapitre. Ensuite, nous présenterons la TEI qui se base sur la norme SGML et qui est dédiée au codage de texte.

2.1 SGML

2.1.1 Introduction

SGML (*Standard Generalized Markup Language)* est une norme permettant à tout type d'information d'être échangé entre des systèmes de manipulation d'informations dissemblables.

Il a pour objet le marquage structurel de documents électroniques, reconnu par l'ISO en octobre 1986. Il est, très vite, devenu populaire suite à son acceptation rapide dans le monde de l'édition, par de grandes compagnies multinationales, par des organisations gouvernementales et, plus récemment, par l'omniprésence de HTML (*Hyper Text Markup Language*) au travers *d'Internet.*

SGML est un outil permettant la manipulation de grandes quantités d'informations électroniques grâce à la structure et à la sémantique qu'il peut leur apporter. En tant que langage de structuration, SGML possède quelques avantages :

- la qualité de présentation du document source est améliorée puisqu'il est clairement structuré ;
- le document peut être utilisé plus rationnellement ;
- les coûts de publication sont réduits (supports électroniques) ;
- les informations peuvent être facilement réutilisées ou encore tenues à jour ;
- la manipulation automatique du document est possible d'un point de vue sémantique.

2.1.2 Principes de base

SGML est donc une méthode standard pour représenter l'information contenue dans un document indépendamment du système utilisé. SGML ne se définit pas comme un langage de marquage de texte en tant que tel, mais il fournit un cadre de travail pour construire différents types de langage de marquage. SGML est donc un métalangage. SGML, à l'aide de la notion de DTD[17], définit les balises permises, requises et la manière de représenter ces dernières.

On peut dégager trois piliers de la philosophie de SGML :

- un langage de marquage descriptif ;
- le principe de description de type de documents ;
- l'indépendance par rapport aux plates-formes utilisées.

A) Langage descriptif

SGML est un langage de marquage descriptif, par opposition à un langage de marquage procédural. Il utilise des balises qui vont faire ressortir la structure du texte, en mettant en évidence, par exemple, le début ou la fin d'un paragraphe. A contrario, un langage procédural insérerait des appels de fonction dans le texte et appellerait, par exemple, la fonction PARAGRAPHE avec comme arguments x, y et z.

Un langage de marquage descriptif permet, lors de la manipulation d'un texte ou d'une partie de texte, d'utiliser un code totalement différent de celui utilisé pour un langage de marquage procédural. En effet, ce code sera séparé du texte, et en conséquence, il offrira la possibilité d'appliquer, en fonction du contexte, de multiples fonctionnalités sur une même partie de texte.

B) Description de type de document

La structuration d'un texte selon un marquage logique (*logical markup*) se fait en 2 étapes :

1. La première étape est la définition d'un ensemble de balises, identifiant tous les éléments d'un document et la définition des règles formelles précisant les

[17] *Document Type Description*

relations entre les différents éléments et leur structure. Ces définitions sont rassemblées dans la DTD (Document Type Definition).

2. La seconde étape est l'insertion du marquage dans le document source suivant les règles définies dans la DTD.

Plusieurs documents peuvent appartenir au même type de documents ; dans ce cas, ils sont décrits dans une même DTD, c'est-à-dire qu'ils ont la même structure logique.

Article A	Article B
Title	*Title*
Section 1	*Section 1*
Subsection 1.1	*Subsection 1.1*
Subsection 1.2	*Subsection 1.2*
Section 2	*Subsection 1.3*
Section 3	*Section 2*
Subsection 3.1	*Subsection 2.1*
Subsection 3.2	*Subsection 2.2*
Subsection 3.3	*Bibliography*
Subsection 3.4	
Bibliography	

Tableau 2-1 - Documents de structures spécifiques différentes mais de même structure logique

Les articles A et B (Tableau 2-1) ont une structure spécifique différente, mais leur structure logique est identique. Il s'agit d'une structure du type suivant : un document est composé d'un titre, suivi par une ou plusieurs section(s) qui elle(s) même(s) est (sont) divisée(s) en zéro ou plusieurs sous-section(s) et il se termine par une bibliographie.

Donc, pour définir la structure formelle de tous les documents, il est nécessaire de construire une DTD commune.

C) Indépendance

Nous avons déjà noté que SGML était indépendant sur un plan abstrait de la plate-forme hôte, de deux manières : la première est l'utilisation d'un langage de marquage descriptif plutôt que procédural ; la seconde est l'emploi du système de DTD.

Pour être complet, il reste à montrer qu'au niveau physique, il l'est aussi. SGML définit une procédure générale, permettant la substitution de chaînes de caractères. SGML est également doté d'une procédure garantissant la consistance de la nomenclature quel

que soit le système hôte, ce qui rend également possible la gestion des jeux de caractères différents selon le système[18].

2.1.3 Structure SGML

SGML est un métalangage qui décrit des règles permettant la définition des systèmes de balises pour chaque type de texte. Les unités textuelles, vues comme unités structurelles, sont appelées des **éléments**.

Les éléments sont encadrés par des **balises** ouvrantes et fermantes. La balise fermante se distingue de la balise ouvrante par une barre oblique « / » placé juste avant le *tag*. Le ***tag*** est le nom permettant de distinguer les balises entre elles. Dans l'exemple suivant, le *tag* est le mot « mabalise ».

```
<mabalise>... élément du texte... </mabalise>
```

Tableau 2-2- Exemple de balise SGML

Ces balises peuvent contenir des attributs qui fournissent une description de l'élément concerné et ils se placent a l'intérieur de la balise ouvrante.

```
<mabalise attribut=valeur>... élément du texte... </mabalise>
```

Tableau 2-3 - Exemple d'attribut d'une balise SGML

L'inclusion d'éléments dans un autre élément est réglementée par la DTD. On définit, au sein de celle-ci, les balises autorisées, ainsi que les conditions dans lesquelles on peut les introduire.

Le Tableau 2-4 [GETEI, 97] présente une anthologie[19] qui ne prend en compte que des poèmes. Chaque poème est composé d'un titre, de strophes et de vers.

[18] Par exemple la chaîne « &eacute » sera remplacée lors de l'affichage par un é.

[19] une anthologie est un recueil de morceaux choisis d'oeuvres littéraires ou musicales.

```
<anthologie>
        <poème><titre>The SICK ROSE</titre>
                <strophe>
                        <vers>O Rose thou art sick.</vers>
                        <vers>The invisible worm</vers>
                        <vers>That flies in the night</vers>
                        <vers>In the howling storm:</vers>
                </strophe>
                <strophe>
                        <vers>Has found out thy bed</vers>
                        <vers>Of crimson joy:</vers>
                        <vers>And his dark secret love</vers>
                        <vers>Does thy life destroy.</vers>
                </strophe>
        </poème>
                        <!-- autres poèmes -->
</anthologie>
```

Tableau 2-4 - Codage SGML d'une anthologie de poèmes

Les éléments à retirer de cet exemple sont :

- une anthologie est composée uniquement de poèmes ;
- un poème a toujours un titre unique qui précède la première strophe ;
- les seuls éléments d'un poème sont le titre et les strophes ;
- une strophe est composée uniquement de vers et un vers est toujours contenu dans une strophe ;
- une strophe ne peut être suivie que par une autre strophe ou par la fin du poème ;
- un vers ne peut être suivi que par un autre vers ou par une nouvelle strophe ;
- la ligne `<!-- autres poèmes -->` représente un commentaire SGML.

On peut se rendre compte que les marquages de fin de titre, de strophes et de vers ne doivent pas être explicites. Le marquage de fin de titre est sous-entendu par le début de la première strophe. Le marquage de fin de poème est sous-entendu par le marquage de début du poème suivant ou par le marquage de fin de l'anthologie. De même, le marquage de fin d'un vers est sous-entendu par le marquage de début du vers suivant ou par le marquage de fin de la strophe.

Lorsque l'on retire les balises redondantes, le codage de l'anthologie se présente comme dans le Tableau 2-5.

```
<anthologie>
        <poème><titre>The SICK ROSE
                <strophe>
                        <vers>O Rose thou art sick.
                        <vers>The invisible worm
                        <vers>That flies in the night
                        <vers>In the howling storm:
                <strophe>
                        <vers>Has found out thy bed
                        <vers>Of crimson joy:
                        <vers>And his dark secret love
                        <vers>Does thy life destroy.</vers>
                        <!-- autres poèmes -->
</anthologie>
```

Tableau 2-5 - Codage SGML simplifié de l'anthologie

2.1.4 Document Type Description

Une DTD est décrite dans un langage défini par le standard SGML. Elle identifie tous les éléments qui sont permis dans un document appartenant au type décrit. SGML permet d'associer à chaque type de texte une DTD. Celle-ci indique les balises autorisées, leurs occurrences et leurs agencements. La DTD donne un nom à chaque élément structurel et, si nécessaire, associe un ou plusieurs attributs à chaque élément et décrit les relations entre éléments.

Le concepteur d'une DTD, et donc d'un type de document, pourra préciser une structure très stricte ou non. Il devra faire la part entre la facilité de suivre de simples règles et la complexité de manipulation de textes réels. Il est important de se rappeler que toutes ces définitions de types de textes sont des interprétations des documents marqués. Il n'existe pas de DTD qui puisse englober toutes les interprétations d'un texte. Cependant, il peut être intéressant d'en privilégier une par rapport à une autre pour une analyse particulière.

Ainsi, la DTD suivante est associée à l'anthologie présentée dans le Tableau 2-5.

```
<!ELEMENT anthologie    - -   (poème+)>
<!ELEMENT poème         - -   (titre?, strophe+)>
<!ELEMENT titre         - o   (#PCDATA) >
<!ELEMENT strophe       - o   (vers+) >
<!ELEMENT vers          o o   (#PCDATA) >
```

Tableau 2-6 - DTD associée à l'anthologie

Cette DTD contient cinq déclarations. Une déclaration se compose de trois parties: un nom ou groupe de noms, deux caractères spécifiant les règles de minimisation et le *content model.*

La première partie de la déclaration est l'identifiant de l'élément qui est déclaré.

La deuxième partie détermine si les balises de début et de fin doivent être présentes. Le premier caractère représente la balise de début et le second la balise de fin. Le symbole « - » indique que la balise doit être présente tandis que la lettre « o » indique que la balise est optionnelle.

Le *content model* spécifie les éléments que peut contenir l'élément qui fait l'objet de la déclaration. Le contenu peut être tantôt d'autres éléments, tantôt des mots réservés. Ainsi, le mot réservé « #PCDATA » (abréviation pour *parsed character data*) signifie que l'élément déclaré peut contenir n'importe quel caractère.

Le Tableau 2-7 nous donne l'interprétation de la DTD du Tableau 2-6.

Identifiant	Balise ouvrante	Balise fermante	Content model
anthologie	obligatoire	obligatoire	une anthologie est composée de un ou plusieurs poèmes
poème	obligatoire	obligatoire	un poème est composé d'un titre optionnel et ensuite d'une ou plusieurs strophes (la virgule donne la notion d'ordre)
titre	obligatoire	optionnelle	un titre est composé d'une suite de caractères
strophe	obligatoire	optionnelle	une strophe est composée de un ou plusieurs vers
vers	optionnelle	optionnelle	un vers est composé d'une suite de caractères

Tableau 2-7 - Interprétation de la DTD de l'anthologie

Quelques exemples plus complexes :

```
<!ELEMENT poème - o (titre?,(strophe+ | couplet+ | vers+) )>
<!ELEMENT poème - o (titre?,(strophe | couplet | vers)+ )>
```

Tableau 2-8 - Eléments complexes

Dans le premier cas, un poème est constitué d'un titre optionnel, suivi d'une ou plusieurs strophes ou d'un ou plusieurs couplets ou encore d'un ou plusieurs vers.

Dans le second cas, un poème est aussi constitué d'un titre optionnel. Cependant, étant donné que le symbole « + » porte sur tout le groupe d'éléments plutôt que sur chaque élément, le titre sera suivi d'un mélange de strophes, de couplets et de vers.

```
<!ELEMENT refrain      - - (#PCDATA | vers+)>
<!ELEMENT poème        - o (titre?,
                            | ( (vers+)
                            | (refrain?, (strophe, refrain?)+ )))>
```

Tableau 2-9 - Un élément complexe imbriqué

Selon cette déclaration, un poème est composé d'un titre optionnel suivi, soit par une suite de vers, soit par un groupe d'éléments. Ce groupe d'éléments est constitué tout d'abord d'un refrain optionnel suivi d'une ou plusieurs occurrences d'un autre groupe composé d'une strophe et d'un refrain optionnel.

Les séquences *refrain - strophe - strophe - refrain* et *strophe - refrain - refrain - strophe* sont des exemples corrects.

Par contre, la séquence *refrain - refrain - strophe - strophe* est incorrecte.

SGML nous permet également de définir pour chaque élément une liste d'attributs.

Un attribut décrit une information spécifique à un élément qui ne concerne pas son contenu. Il est intéressant, par exemple, d'avoir un attribut qui identifie les différentes occurrences d'un élément pour pouvoir, dans un texte, se référer à un élément particulier.

La DTD du Tableau 2-11 nous permet d'utiliser des attributs ainsi que nous le présentons dans le Tableau 2-10.

```
<poème id=P1 status="draft"> ... </poème>
```

Tableau 2-10 - Balise contenant des attributs

```
<!ATTLIST poème
          id      ID                                  #IMPLIED
          status (draft I revised I published)        draft   >
```

Tableau 2-11 - Définition d'attributs pour l'élément poème

La chaîne de caractères ATTLIST indique la présence d'une liste d'attributs. La déclaration d'une liste d'attributs est décomposée en trois parties.

La première partie fournit le nom de l'élément concerné par la liste d'attributs.

La deuxième partie énumère les valeurs que peut prendre l'attribut. Cette partie peut être composée, soit de mots-clés SGML qui définissent les différentes valeurs possibles pour l'attribut, soit d'une liste de valeurs possibles définies par l'utilisateur.

Valeur	Fonction
CDATA	La valeur de l'attribut peut être toute chaîne de caractères.
ID	La valeur de l'attribut est un identifiant unique de l'élément.
IDREF	La valeur de l'attribut contient un pointeur vers un autre élément. Cet autre élément doit avoir un attribut identifiant « id » dont la valeur n'est pas nulle.
NMTOKEN	La valeur de l'attribut doit être une chaîne de caractères alphanumériques.
NUMBER	La valeur de l'attribut doit être une suite de chiffres.

Tableau 2-12 - Valeurs prédéfinies pour les attributs

La troisième partie associe à l'attribut sa valeur par défaut. Celle-ci indique comment interpréter l'absence, dans le texte SGML, de l'attribut concerné. De nouveau, il existe deux méthodes pour définir l'attitude à adopter : par l'utilisation de mots-clés SGML, ou par la définition d'une valeur spécifique.

Valeur	Fonction
#REQUIRED	Une valeur doit être associée à l'attribut, sinon l'élément est déclaré incorrectement balisé.
#IMPLIED	Si aucune valeur n'est associé à l'attribut, l'attribut est considéré comme étant sans valeur.
#CURRENT	Si aucune valeur n'est associée à l'attribut, la dernière valeur utilisée pour cet attribut lui est donnée.

Tableau 2-13 - Types présences d'un attribut

L'analyse du Tableau 2-11 (définition d'attributs) révèle que:

- L'élément « poème » est concerné par la liste d'attributs.

- L'attribut « `id` » fournit pour chaque poème une valeur identifiante. Cependant, suite à la valeur par défaut « #IMPLIED », l'utilisateur n'est pas obligé de fournir un identifiant pour chaque poème, mais uniquement pour ceux auxquels il compte faire référence. Les poèmes dénués de cet attribut seront considérés tout simplement sans identifiant.
- L'attribut « `status` » peut prendre l'une des trois valeurs : *draft*, *revised* ou *published* . Dans ce cas, la valeur par défaut est *« draft »*.

2.2 La *Text Encoding Initiative*[20]

2.2.1 Introduction

La *Text Encoding Initiative* est un projet international qui a pour objectif la mise au point d'un ensemble de normes pour la préparation et l'échange de textes électroniques. La TEI est née lors d'une réunion organisée en novembre 1987 au *Vassar College (New-York)* à laquelle ont participé diverses personnalités travaillant dans le domaine de l'archivage, de la structuration ou de l'analyse des textes électroniques. A l'époque, la grande variété des formats de codage et de représentation des textes, quasiment tous incompatibles, constituait un obstacle majeur à l'échange des données et à la recherche. Les chercheurs présents à *Vassar* sont donc tombés d'accord sur la nécessité de travailler à la définition d'un nouveau format de codage des textes électroniques et en ont posé les principes de base :

- être aussi complet que possible ;
- être simple, clair et concret ;
- être facile à utiliser sans logiciel particulier ;
- être rigoureusement défini ;
- permettre un traitement efficace ;
- être ouvert à des extensions définies par l'utilisateur.

Les recommandations s'adressent à tous ceux qui souhaitent échanger des informations stockées sous forme électronique. L'accent est mis sur l'échange de données textuelles mais d'autres types de données comme les images et les sons sont aussi pris en compte. Les recommandations peuvent être appliquées aussi bien pour créer de nouvelles informations que pour échanger des informations existantes.

[20] [TEI, 97]

La TEI tire donc son origine, d'une part, de l'anarchie qui régnait dans la communauté scientifique en matière de format et, d'autre part, du nombre croissant de traitements que les chercheurs effectuent sur les textes sous forme électronique.

Au même titre que HTML, utilisé sur le *Web* par un grand nombre de personnes, la TEI est une application de la norme SGML, une DTD conséquente. Elle contient plus de 500 éléments (50 pour HTML) et autant d'attributs ; c'est dire si les formes textuelles susceptibles d'être codées à l'aide de la TEI sont nombreuses et variées. Il en va ainsi du roman, de la poésie, du théâtre, de l'article scientifique ou technique, mais également du dictionnaire, du dialogue multimodal (parole et geste), etc. A cet égard, il est important de noter que, contrairement à HTML qui a un objectif de présentation (de l'encre sur une page !), la TEI normalise l'étape de représentation des données.

Le but de la TEI est de faciliter l'échange et la circulation des documents électroniques au sein de la communauté scientifique. Les différents équipements informatiques ainsi que les différents types de codage constituent souvent un obstacle à l'échange de données. La TEI a pour but d'améliorer cette situation en proposant des conventions de codage neutres par rapport aux équipements.

Les recommandations de la TEI s'appuient sur la norme SGML. Elles contiennent un ensemble de DTD (*Definition Type Document*) accompagnées de commentaires très détaillés sur l'usage des balises et des attributs. La TEI contient un grand nombre de modules adaptables selon le type de texte sur lequel on travaille.

La DTD TEI est construite de façon modulaire. Ainsi, elle contient :

- Un jeu de balises «noyau » (core tag set) composé d'éléments communs à tous les types de textes (divisions, paragraphes, etc.).
- Des ensembles de balises de base (base tag sets) pour chaque type particulier de texte (prose, poésie en vers, etc.).
- Des jeux de balises additionnelles (additional tag sets) pour des mécanismes particuliers qui peuvent se superposer à n'importe quel type de texte (liens *hypertextuels*, etc.).

Les balises du noyau sont toujours présentes, mais les jeux de balises de base ou additionnelles sont «chargées » dans la DTD en fonction des besoins des utilisateurs. Un ensemble de mécanismes est aussi fourni pour permettre aux utilisateurs de rajouter leurs propres balises ou les balises existantes sans avoir à réécrire la DTD.

2.2.2 Principes de la TEI

La TEI est un codage qui, sans être universel, se veut tout de même assez général pour intéresser le plus grand nombre possible de chercheurs, éditeurs, etc. Disons dès à présent que certains points ne sont pas, ou peu, abordés aujourd'hui par la TEI, par exemple tout ce qui touche au contenu des images et au son.

Ce codage se veut indépendant des «plates-formes », c'est-à-dire indépendant de telle ou telle marque d'ordinateur, de tel ou tel système commercial de manipulation de texte, mais aussi de tel ou tel réseau.

Ce codage est un codage interne. Ce n'est pas obligatoirement la forme sous laquelle les chercheurs doivent l'utiliser. En particulier, divers types d'outils (ou diverses fonctionnalités d'un même produit) sont à employer pour travailler confortablement avec la TEI (en ignorant les balises).

2.2.3 Structure générale d'un document TEI

Un texte conforme à la TEI comprend:

1. un en-tête (l'élément `<teiHeader>`)
2. transcription du texte lui-même (l'élément `<text>`)

A) Codage de l'en-tête d'un document

Le *header* est un ensemble d'éléments SGML qui fournit des informations sur le texte et sur son codage. Il fournit les données nécessaires au bibliothécaire qui veut cataloguer le texte, à l'étudiant qui l'utilise ou, encore, au programme d'ordinateur qui effectue des manipulations sur ce texte.

Au départ, la TEI a proposé de diviser le *header* en trois parties. Nous verrons que cette proposition a été quelque peu modifiée pour permettre un traitement harmonisé de presque tout les types de textes. En effet, dans un premier temps, la TEI s'était uniquement penchée sur le problème du traitement électronique des textes écrits tels que livres, articles, etc. Depuis lors, le besoin de coder des textes d'origine parlée s'est fait ressentir.

Observons le *header*[21] issu de la proposition initiale:

[21] Dans le cas d'un corpus de textes, il existe un *header* qui contient les éléments communs au corpus tout entier et des *headers* individuels pour chaque texte du corpus.

- La partie *file description* (<filedesc>) contient la description bibliographique du texte électronique. On y trouve notamment le titre de l'oeuvre et ses auteurs (<titlestmt>) ou encore des informations sur la publication et la distribution du texte, et sur le texte source (<publicationstmt>).
- La partie *encoding description* (<encodingdesc>) documente les principes d'édition utilisés dans la transcription du texte, tels que le traitement des citations, la segmentation du texte, etc.
- La partie *revision description* (<revisiondesc>) permet de tenir à jour un historique des changements effectués dans le document.

Le Tableau 2-14 met en évidence les éléments du header présentés ci-dessus.

```
<teiheader>
  <filedesc>
   <titlestmt>
    <title>short title: an electronic edition</title>
    <author>name of the original author, if known, otherwise`Anon'
          </author>
    <editor>name of the editor of the edition from which this copy was
taken</editor>
   </titlestmt>
   <publicationstmt>
    <publisher>Thesaurus Linguarum Hiberniae</publisher>
    <address>
     <addrline>University College Cork</addrline>
     <addrline>Royal Irish Academy</addrline>
    </address>
    <date>Date of publication</date>
    <idno>ID number allocated by the Editorial Board</idno>
   </publicationstmt>
   <sourcedesc>
    <biblfull>
     <titlestmt>
      <title>Full title of the original work</title>
      <author>Name of original author</author>
     </titlestmt>
     <editionstmt>
      <edition>Electronic edition prepared for the Thesaurus
          Linguarum Hiberniae project of the Royal Irish Academy and
          University College Cork (CURIA).</edition>
     </editionstmt>
     <publicationstmt>
      <publisher>Thesaurus Linguarum Hiberniae</publisher>
```

```
        </publicationstmt>
       </biblfull>
     </sourcedesc>
   </filedesc>
   <encodingdesc>
     <editorialdecl>
       <quotation>
         <p>Details of how quotations are handled.</p>
       </quotation>
       <segmentation>
         <p>Details of how segmentation is handled.</p>
       </segmentation>
       <p>Any further comments about editorial method.</p>
     </editorialdecl>
   </encodingdesc>
   <revisiondesc>
     <change>
       <date>date of changes</date>
       <respstmt>
         <name>name of person</name>
         <resp>responsibility</resp>
       </respstmt>
       <item>what was changed</item>
     </change>
   </revisiondesc>
 </teiheader>
```

Tableau 2-14 - Exemple de header relativement complexe

B) Codage du corps du document

Le Tableau 2-15 et le Tableau 2-16 présentent les éléments permettant de structurer les corps du texte (balise <text>).

Balise	Fonction
<front>	ensemble de tous les éléments (en-têtes, page de titre, préfaces, dédicaces, etc.) situés avant le début du texte lui-même.
<group>	rassemblement de plusieurs textes unitaires ou groupes de textes.
<body>	cette balise situe le corps entier d'un texte unitaire seul, à l'exclusion de toute pièce liminaire ou annexe.
<back>	ensemble de toutes les annexes qui suivent le texte principal.

Tableau 2-15 - Balises composant un texte TEI

```
<TEI .2>
<teiHeader> [informations contenues dans l'en-tête TEI]
</teiHeader>,
<text>
   <front>[ textes préliminaires... ] </front>
   <body>[ corps du texte... ] </body>
   <back> [annexes.. ] </back>
 </text>
</TEI.2>
```

Tableau 2-16 - Exemple de corps de texte TEI

Le texte lui-même, compris dans la balise <body>, peut être subdivisé en parties. Le Tableau 2-17 présente les subdivisions possibles.

Balise	Fonction
<p>	Délimitation des paragraphes (Tableau 2-19).
	Cette subdivision contient des pièces liminaires, le corps ou des annexes d'un texte.
<div1>	Regroupement de premier niveau des pièces liminaires du corps ou des annexes d'un texte. Si une découpe plus fine que <div1> est nécessaire, <div1> peut être divisé en éléments <div2>, et un <div2> en éléments <div3> et cela jusqu'au niveau 7. Au-delà de 7 niveaux, il sera nécessaire de modifier l'ensemble du balisage TEI pour lui permettre d'accepter <div8> ou encore d'employer un élément non numéroté dont le degré d'imbrication est illimité.

Tableau 2-17 - Eléments subdiviseurs d'un texte

Tous ces éléments contiennent trois attributs :

Attribut	Fonction
type	Cet attribut indique le type de division de l'élément qui le contient. Sa valeur peut être livre, chapitre, poème, div,...
id	id fournit un identifiant unique, propre à l'élément qui le contient. Il servira dans le cas des références croisées ou d'autres liens pointant vers cette division.
N	Il indique un nom court, un mnémonique ou un numéro pour l'élément. Il servira à identifier un autre élément dont l'attribut id a la même valeur que N.

Tableau 2-18 - Attributs des éléments subdiviseurs d'un texte

```
<body>
<p id=ch1p1>Nous apprécions beaucoup les résultats qui
ont été obtenus et nous en tiendrons compte. Il faut
toutefois garder en mémoire...
</p>
</body>
```

Tableau 2-19 - Utilisation de la balise <p>

2.2.4 le codage d'un document parlé

A côté du modèle existant pour les textes écrits, la TEI propose des conventions de codage de textes parlés, compatibles avec les propositions précédentes. Celles-ci, sans déformer les caractéristiques du *speech,* tentent de satisfaire les besoins des différents utilisateurs du *spoken text.*

A) En-tête d'un document parlé

Etant donné que le *speech* est beaucoup plus dépendant du contexte qu'un texte écrit, il est nécessaire de le documenter davantage. On veut y associer des renseignements tels que : le degré de spontanéité ou de préparation, les participants, le cadre du *spoken text* et le matériel d'enregistrement et de transcription.

Ainsi, la TEI proposait initialement de diviser le *header* d'un *spoken text* en trois sections:

- une section script (pour les textes préparés à l'avance) ;
- une section donnant les détails sur l'enregistrement ;
- une section donnant les détails sur le transcripteur et sur les principes de la transcription.

On peut remarquer que, dans cette proposition, rien ne concerne le cadre et les participants. Il était donc nécessaire de proposer des éléments additionnels dans le *header* des *spoken texts.*

Cette proposition a été modifiée afin d'harmoniser le traitement des *spoken* et *written texts.* Le *header* des documents parlés (Tableau 2-20) est composé des mêmes balises que celui des textes écrits. Cependant, la balise `<profiledescription>` concernant les participants a été ajoutée.

Balise	Fonction
`<filedescription>`	Si la transcription existe seulement sous une forme

	électronique, elle sera documentée dans cette balise.
<sourcedescription>	S'il y a une transcription écrite existante, celle-ci sera documentée dans cette balise. Les sections script et enregistrement de la proposition initiale deviennent des éléments optionnels de cette balise.
<encodingdescription>	Les principes de transcription seront placés dans cette balise.
<profiledescription>	Cette balise fournit une description détaillée des aspects non bibliographiques du texte. Elle décrit le langage utilisé, le contexte du document ou encore les participants. Il existe un large éventail de paramètres suivant le type de spoken ou written text concerné.

Tableau 2-20 - Eléments du header TEI adaptés aux documents parlés

Les quatre balises citées dans le Tableau 2-20 forment un *header* standard valable pour tous les types de textes, aussi bien écrits que parlés. Le contenu de ces balises est adapté grâce à un grand nombre d'éléments optionnels qui s'appliquent selon le type de texte (écrit ou parlé).

Au delà de l'aspect du *header,* les *spoken texts* réclamaient de nouveaux *tags.* En effet, les *written texts* sont divisés en parties aisément reconnaissables telles que des chapitres, des sections, des paragraphes, etc. Par contre, les *spoken texts,* eux, ne sont pas divisés aussi clairement. Il était donc indispensable d'y apporter une certaine structure pour les manipuler. Les éléments contenus dans le Tableau 2-21 furent donc ajoutés pour les besoins de segmentation, et ceux du Tableau 2-22 pour permettre le codage d'éléments spécifiques .

Balise	Fonction
<text>	Un spoken text marqué par la balise <text> est une version électronique d'un speech qui: • forme un tout ; • peut être décrite par un seul header ; • représente une tranche de temps sans discontinuité signifiante.
	Cette balise représente une subdivision d'un spoken text, comprenant une ou plusieurs utterance[22](s), traitée(s) comme une unité bien définie. Il peut y avoir plusieurs niveaux d'imbrication de .
<u>	Cette balise représente le tour de parole qui est habituellement précédé et suivi d'un silence ou du changement de locuteur. Les utterances peuvent varier très fortement en longueur. Une utterance peut être constituée de tout un discours ou d'un seul mot. Les balises <u> ne concernent qu'un locuteur contrairement aux balises <text> et .
<s>	il s'agit de la subdivision d'une utterance selon des critères prosodiques ou syntaxiques.

Tableau 2-21 - Eléments de segmentation d'un spoken text

Balise	Fonction
<pause>	Cette balise représente une pause entre ou à l'intérieur de tours de parole.
<vocal>	Cette balise représente tout phénomène non lexical entre ou à l'intérieur d'une utterance.
<kinesic>	Cette balise représente les phénomènes non vocaux mais communicatifs comme les gestes.
<event>	Cette balise représente les phénomènes non vocaux, non communicatifs comme un bruit accidentel.
<writing>	Un texte écrit peut apparaître comme un élément important dans certaines situations. Il en est ainsi lors d'une lecture ou d'une émission de TV. Un attribut who identifie la personne qui présente ou crée l'écrit tandis qu'un attribut type caractérise l'écrit.

Tableau 2-22 - Eléments spécifiques aux textes parlés

[22] La balise <u> représente un tour de parole.

2.3 Conclusion

La norme SGML et sa DTD TEI sont très couramment utilisés au sein de l'équipe Dialogue.

Les éléments de ce chapitre permettront de lire et de comprendre la structure d'un document (et une DTD) SGML ou TEI. Ces éléments sont le fondement même des manipulations que l'on peut faire sur un document SGML. Ils forment les unités de base sur lesquelles travaillent les applications que nous allons présenter dans le chapitre suivant.

3. Applications particulières

Nous nous attarderons dans ce chapitre à présenter le contexte dans lequel nos applications ont été développées. La première application a pour objet l'aide à l'analyse de dialogue. La seconde permet l'édition (graphique) de documents SGML sous forme d'arbres.

Le travail de recherche mené sur le codage de dialogues se base sur le projet GOCAD. Comme nous le verrons, ce projet a donné naissance à un corpus de dialogues. Les travaux concernant les différents niveaux de codage pour un même dialogue et le codage des références et coréférences dans les dialogues homme-machine réalisés par Florence Bruneseaux ont pour fondement ce corpus GOCAD.

La seconde application a été développée dans l'idée de satisfaire à deux besoins : le premier est la réalisation d'une interface graphique permettant de manipuler un document SGML sous la forme d'un arbre ; le second est de permettre, à l'aide de la même application, de manipuler des arbres adjoints, notion qui sera expliquée dans ce chapitre.

3.1 Dialogue homme-machine et TEI

Les interfaces homme-machine sont caractérisées par une relation directe entre l'utilisateur et la machine et supposent une très bonne connaissance du système employé. Toute action suppose l'apprentissage de manipulations adéquates pour arriver au but escompté. En effet, l'utilisateur , pour agir sur la tâche, doit effectuer une suite d'actions au moyen du clavier, de la souris ou d'autres périphériques. La conception d'un système plus souple avec le langage naturel comme moyen de communication entre l'homme et la machine permettrait de s'affranchir de ces contraintes de manipulation parfois très complexes. L'homme utiliserait les machines non plus comme objets mais comme partenaires dans la réalisation de sa tâche. L'utilisateur donnerait ses ordres, transmettrait ses intentions à la machine afin d'agir sur l'application. Le système ne se contenterait pas seulement d'exécuter une requête mais deviendrait un assistant pour l'utilisateur, soit en le conseillant, soit en le critiquant ou même parfois en décidant à sa place.

3.1.1 Présentation du projet GOCAD

Le projet GOCAD [BRUNE, 96a] a été lancé par le département informatique de l'Ecole Nationale Supérieure de Géologie (ENSG) en 1989. Le but de ce projet est de développer une nouvelle approche de modélisation d'objets géologiques assistée par ordinateur. GOCAD est un logiciel de modélisation de surfaces de type géologique développé par le CRIN (l'équipe Infographie). Alors que GOCAD était développé initialement pour être appliqué au domaine pétrolier, les concepteurs se sont également tournés vers le domaine médical.

Les testeurs du logiciel, huit étudiants de deuxième année de géologie à l'ENSG, avaient pour but de réaliser quatre tâches de difficultés croissantes s'intègrant les unes dans les autres pour réaliser une coupe géologique.

Deux personnes, appelées compères, ayant des compétences dans le domaine géologique et maîtrisant parfaitement le logiciel, avaient pour rôle de simuler le fonctionnement de l'interface. Ils devaient agir comme une « interface intelligente » c'est-à-dire qu'ils devaient non seulement exécuter les ordres du sujet (étudiant) mais aussi redéfinir les objectifs à atteindre en priorité. Ils devaient donc vérifier la pertinence des demandes du sujet. La grande différence entre sujets et compères vient du fait qu'ils ne possèdent pas des compétences égales. Les premiers ont une bonne connaissance en géologie mais quelques notions en informatique. Les compères, quant à eux, sont parfaitement familiarisés avec l'application GOCAD. Le dialogue doit permettre la combinaison des connaissances de chacun en vue de réaliser la tâche demandée.

Dans l'expérimentation, pour atteindre l'objectif, l'utilisateur peut:

- soit agir directement sur la tâche par l'utilisation de la souris ;
- soit agir sur la tâche par l'intermédiaire d'un système de dialogue et donc du langage naturel.

Le système de dialogue n'existant pas, c'est un compère qui tient ce rôle. Les dialogues sujet-compère ont été construits selon la méthode du « Magicien d'Oz ». Celle-ci consiste à faire simuler par un compère, à l'insu des sujets une machine capable de réaliser des activités humaines complexes, comme comprendre un langage humain sans restriction.

La Figure 1 représente le traitement d'une requête de l'utilisateur dans le système GOCAD. Si l'utilisateur agit directement sur l'application par l'intermédiaire de la souris, le résultat de la manipulation sera immédiatement visible à l'écran. Si l'utilisateur agit par

l'intermédiaire du langage naturel, le retour visuel associé à la requête sera accompagné par un message du compère.

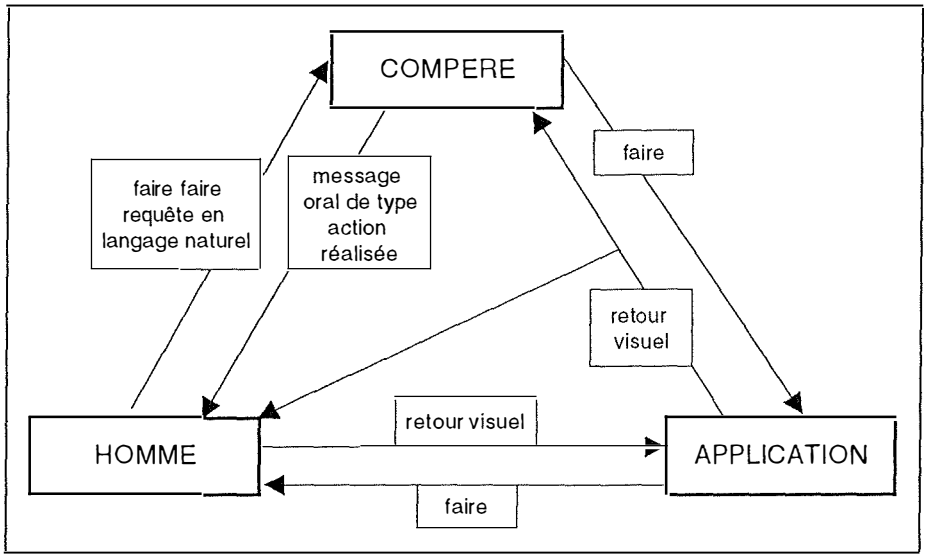

Figure 1 - Relations entre un utilisateur et une application

3.1.2 Transcription initiale des dialogues

Pour procéder à une étude approfondie des dialogues Homme - Machine, il était nécessaire d'établir des conventions de représentation. Celles-ci ont pour but de symboliser toutes les situations qui peuvent surgir lors d'un dialogue (Tableau 3-23). Ainsi, elles permettent de déterminer le locuteur qui énonce la phrase, les temps d'attente ou encore l'utilisation de la souris.

Codage	Interprétation
<who = X>	Désignation de la personne (X= sujet ou compère) énonçant la phrase.
#	Début de commentaire.
O	Courte pause (5 secondes) entre la fin d'une phrase du compère et le début de la réponse du sujet.
Oo	Attente de 10 à 20 secondes.
Ooo	Pause de plus de 20 secondes.
/-/	Pause de moins de 5 secondes au milieu d'une phrase du sujet.
/	Le sujet est coupé dans sa phrase par le compère.
(...)	Séquence non reconnue par le transcripteur.
(abcd)	Séquence dont le transcripteur n'est pas certain.
(abcd #xyz)	Séquence dont le transcripteur n'est pas certain. Le commentaire qui suit permet de donner la cause de cette incertitude.
<CLICKB # commentaire>	Le sujet a cliqué sur un des boutons du pilote de caméra. Le commentaire permet de préciser sur quel bouton il a cliqué.
<CLICKO # commentaire>	Le sujet a cliqué dans la caméra. Le commentaire permet de dire si le sujet a désigné un objet ou a cliqué ailleurs dans la caméra.
<\C>	Ce symbole se rapporte au dernier <CLICKB> ou <CLICKO>. Il signifie que le bouton de la souris est resté enfoncé durant tout le temps séparant les deux symboles.
<CLICKDEP>	Le sujet décrit un mouvement assez long avec la souris en gardant le bouton pressé.

Tableau 3-23: Eléments pour réaliser la transcription initiale

Exemple 1:

74 : **<who = Compère>** Veuillez désigner la partie à supprimer
75 : **<who = Sujet>** Supprimer la partie de **/-/** H2 en dessous de H3
76 : **<who = Compère>** Bien compris
77 : **<who = Compère>** Veuillez désigner la partie à supprimer à la souris
78 : **<who = Sujet>** Revenez au plan de coupe
79 : **<who = Compère>** Bien compris
80 : **<who = Compère>** Opération terminée
81 : **<who = Sujet>** Supprimez **<CLICKO # sur la coupe>** la partie désignée par la souris
82 : **<who = Compère>** Cette vue ne permet pas de désigner la partie de la surface à supprimer
83 : **<who = Sujet> o** Revenir à la vue générale
84 : **<who = Compère>** Bien compris
85 : **<who = Compère>** Opération en cours
86 : **<who = Compère>** Opération terminée
87 : **<who = Sujet>** Supprimez la partie de, euh, la courbe brune à l'intérieur de ce **(...) <CLICKO # intersection>**
88 : **<who = Compère>** Voulez-vous cacher la courbe H3 ?
89 : **<who = Sujet> o <CLICKO #>** Euh, oui
90: **<who = Compère>** Bien compris

Exemple 2:

70: **<who=sujet> <CLICKDEP # surface>** Augmentez la densité du maillage autour de ces points.

Tableau 3-24 - Deux exemples de transcription initiale

Le Tableau 3-24 contient deux exemples de transcription initiale qui est donc le travail préparatoire pour permettre la réalisation d'un codage TEI. Ce travail est subjectif et peut varier d'un transcripteur à l'autre.

L'étude du corpus obtenu à partir de l'expérimentation du logiciel GOCAD a pour but de proposer un codage unique basé sur le format SGML et respectant les directives générales proposées par la TEI pour l'ensemble des dialogues homme-machine.

3.1.3 Codage adopté pour le corpus GOCAD

A) Header

Il existe un *header* pour tout le corpus GOCAD qui regroupe les informations communes à tous les dialogues. Ce *header* est structuré de la manière décrite dans le chapitre 2.

Voici les informations présentes dans les différentes balises du *header* de ce corpus:

- La balise `<filedesc>` contient les informations concernant les caractéristiques de l'enregistrement, c'est-à-dire le lieu, la date, l'équipement, etc. (Tableau 3-25).

```
<filedesc>
<titleStmt>
        <title>corpus GOCAD </title>
</titleStmt>
<sourceDesc>
        <recordingStmt>
                <equipment>
                        Organisation de l'expérience: un compère dans une salle, un
                        sujet dans un autre. Ils communiquent par l' intermédiaire
                        d'une caméra et d' un magnétophone.
                .....
                </equipment>
                <date>8 et 9 novembre 1993 </date>
                <place>CRIN</place>
        </recordingStmt>
</sourceDesc>
</fileDesc>
```

Tableau 3-25 - Extrait de la balise `<filedesc>` du header du corpus GOCAD

- La balise `<encodingdesc>` contient notamment la description du projet, les messages standards définis par les compères ou encore les différents types d'hésitations et leur significations (Tableau 3-26).

<encodingDesc>
<projectDesc>
Réaliser un dialogue de type magicien d'Oz avec une interface multimodale. Dans ce but, les consignes fournies au départ aux sujets sont les suivantes:
Tâche 1 : Modélisation d'une surface à partir d'un ensemble de points donnés
Données : Un ensemble de points stockés dans le fichier Vtop1.vs
Objectif à atteindre : Modéliser une surface qui soit correctement ajustée aux point de données.
.....
</projectDesc>
<editorialDesc>
<segmentation>
L'identification des segments est du type: « id=c1u69seg1 » où sont indiqués: le compère réalisant l'énoncé (c1), le numéro de l'énoncé (u69) et le numéro d'introduction du segment dans l'énoncé (seg1). L'information indiquée sous n correspond à la ligne de référence dans le corpus initial.
</segmentation>
<stdVals>
Les messages standards définis par les compères sont codés comme suit:
message reçu: mc1 (signifie: message de confirmation de type 1)
.....
</stdVals>
</editorialDesc>
</encodingDesc>

Tableau 3-26 - Extrait de la balise <encodingDesc> du header du corpus GOCAD

- La balise <profiledesc> décrit les participants à l'expérience (Tableau 3-27).

```
<profileDesc>
<creation>
        <dateRange from='1993-11-08' to='1993-11-09' >8 et 9 novembre 1993
        </dateRange>
</creation>
<langUsage>
        <language id=FR wsd=wsd.fr>français</language>
</langUsage>
<particDesc>
<person id=S1 sex=U age=young>
.....
<person id=S8 sex=U age=young>
</particDesc>
</profileDesc>
```

Tableau 3-27 - Extrait de la balise <profileDesc> du header *du corpus GOCAD*

- La balise <revisiondesc> est vide étant donné qu'aucune modification n'a été apportée au corpus initial.

Chaque dialogue possède également un *header* reprenant des informations propres à ce dialogue telles que son nom, sa date de création, etc.

B) Codage du dialogue proprement dit

Dans le cadre de DHM[23], le geste et la parole occupent une place capitale puisque l'utilisateur peut en faire usage conjointement dans la réalisation d'une tâche donnée. Un autre élément important est la mise à jour de l'évolution de la tâche.

Trois éléments fondamentaux apparaissent donc lorsque l'on parle de DHM :

- codage du texte proprement dit ;
- codage des gestes ;
- état de la tâche.

Différents niveaux de codage pour un même dialogue ont été proposés par Florence Bruneseaux [BRUNE, 96b].

En ce qui concerne le codage du texte et du geste, une décomposition en niveaux pourra mettre en parallèle dialogue et geste afin de comprendre comment ils se combinent

[23] Dialogue Homme-Machine

dans la réalisation d'une tâche donnée. Le choix du niveau de description dépendra du but pour lequel le codage est réalisé ainsi que du degré de précision désiré.

Ce codage en niveaux s'effectue à partir d'une retranscription initiale du dialogue. Dans le cadre de cette analyse, la transcription initiale est celle d'un dialogue réalisé à partir de l'expérimentation du logiciel GOCAD.

De manière générale, on peut dire que le premier niveau offre un minimum de codage, le deuxième fournit un supplément d'informations et le troisième donne un maximum d'informations. Chaque niveau apporte un degré de précisions supplémentaires par rapport au niveau précédent.

Avant de passer aux principes de codage de la parole et du geste, voyons brièvement en quoi consiste cette décomposition en trois niveaux (Tableau 3-28).

Niveau	Parole	Geste
1	Mise en évidence des tours de parole, des pauses et des notes du transcripteur.	Il permet une description du geste et une présentation de l'objet après le déplacement.
2	De nouveaux attributs sont ajoutés dans les balises déjà définies. De nouvelles balises sont introduites permettant notamment la segmentation du tour de parole.	Il apporte des indications sur le geste (forme, sens, vitesse de déplacement) et donne sa situation temporel par rapport au discours.
3	Il complète le niveau 2 d'un point de vue qualitatif et quantitatif. Du point de vue quantitatif, il apporte des précisions sur la longueur des pauses. Du point de vue qualitatif, il donne les raisons pour lesquelles un segment peut être déclaré impossible à traiter.	Il permet de décomposer un mouvement en éléments de niveau 1.

Tableau 3-28 - Les trois niveaux de codage de la parole et du geste

Le codage traitant de l'état de la tâche sera, quant à lui, abordé plus loin. Nous verrons comment coder les actions du sujet sur l'environnement ainsi qu'une manière de décrire les objets tels qu'ils se présentent au sujet.

1. Codage de la parole

Nous allons maintenant définir plus précisément et illustrer les éléments mis en évidence par le codage de la parole.

a. Niveau 1

Le niveau 1 permet une décomposition élémentaire du dialogue, à savoir les tours de parole, les pauses et les notes du transcripteur.

Tours de parole et identification des locuteurs

Ce niveau 1 regroupe les énoncés successifs d'un même locuteur sous une même balise <u>. Les attributs id et who (Tableau 3-29) sont associés à cette balise.

Attribut	Fonction
id	id est l'identifiant de l'énoncé décrit.
who	who indique le locuteur prenant la parole. Sa valeur est un identifiant représentant le locuteur. L'identité exacte du locuteur est dans le <header> pour les sujets et dans le <front> pour les compères.

Tableau 3-29 - Attributs de la balise <u>

Dans le Tableau 3-30, nous avons trois tours de parole (trois balises <u>). Les attributs id des balises <u> sont formés d'une lettre "u" et d'un chiffre correspondant à la ligne où l'énoncé est retranscit dans la représentation initiale.

Transcription initiale:

```
72 :<who=Compère> Les surfaces n'ont aucun bord commun
73 :<who=Compère> Désirez-vous que je les associe ?
74 :<who=Sujet> Oui
75 :<who=Compère> Vous obtiendrez une surface en quatre
    morceaux
76 :<who=Compère> Opération terminée
```

Représentation adoptée:

```
<u id=u72 who=C1 >
     Les surfaces n'ont aucun bord commun
     Désirez-vous je les associe ?
</u>
<u id=u74 who=S6>Oui</u>
<u id=u75 who=C1 >
     Vous obtiendrez une surface en quatre morceaux
     Opération terminée
</u>
```

Tableau 3-30 - Exemple d'utilisation de la balise <u>

Pause

Le niveau 1 signale également la présence de pauses (Tableau 3-31). Une pause à l'intérieur d'un énoncé est indiquée à l'aide de la balise <pause>. Des indications supplémentaires sur le type de pause pourront être données au niveau 2.

Transcription initiale:

```
19 : <who=compère> Opération terminée
20 : <who=Sujet> Bien /-/ /-/ créer des surfaces /-/ Ah
     pourquoi il change de vue?
21 : <who=Compère> Veuillez répéter
```

Représentation adoptée:

```
<u id=u19 who=C1>Opération terminée</u>
<u id=u20 who=S4>Bien<pause>créer des surfaces<pause>
          Ah pourquoi il change de vue ?
</u>
<u id=u21 who=C1>Veuillez répéter</u>
```

Tableau 3-31 - Exemple de codage d'une pause située au milieu d'un énoncé

Codage d'un commentaire

Un commentaire du transcripteur (les commentaires sont précédés d'un symbole # dans la transcription initiale) est transformé en une balise <note>. Comme ces commentaires sont conservés dans le corps du texte, la balise <note> a comme attribut place qui indique sa position (Tableau 3-32).

Transcription initiale :

```
121 : <who=Sujet> Interpoler la structure # la quatrième
      surface à l'écran
122 : <who=Compère> Message reçu
```

Représentation adoptée :

```
<u id=u121 who=S1 Interpoler la structure <note
place=inline> la quatrième surface à l'écran</note> </u>
<u id=u122 who=C1>Message reçu</u>
```

Tableau 3-32 - Exemple de codage d'un commentaire

b. Niveau 2

Le niveau 2 complète le niveau précédent par l'intermédiaire d'attributs supplémentaires associés aux balises déjà définies. C'est également dans ce niveau que le tour de parole est segmenté ou encore que les transitions entre les énoncés sont traitées.

La segmentation du tour de parole

Les énoncés successifs d'un même locuteur sont regroupés sous une même balise <u> au niveau 1. Cette balise <u> est divisée en *x* segments correspondant au nombre d'énoncés. La balise <seg> décompose le tour de parole en plusieurs phrases et dispose des attributs décrits dans le Tableau 3-33.

Attribut	Fonction
id	Identifiant du segment décrit.
type	Cet attribut caractérise le segment. • pour un segment préencodé: sa valeur est l'identifiant du message standard ; • pour un segment non préencodé: sa valeur est sentence.
n	Valeur attribuée au segment.

Tableau 3-33 - Attributs de la balise <seg>

Les énoncés préencodés sont des messages standards qui sont de types confirmation, attente ou fin. Ces messages ont été définis par les compères avant l'expérimentation. Un identifiant unique a été attribué à chacun de ces messages.

Transcription initiale:

```
72 : <who=Compère> Les surfaces n'ont aucun bord commun
73 : <who=Compère> Désirez-vous que je les associe ?
74 : <who=Sujet> Oui
75 : <who=Compère> Vous obtiendrez une surface en quatre
      morceaux
76 : <who=Compère> Opération terminée
```

Représentation adoptée:

```
<u id=u72 who=C1 >
 <seg id=c1u72seg1 type=sentence n=72> Les surfaces  n'ont
  aucun bord commun</seg>
 <seg id=c1u72seg2 type=sentence n=73>Désirez-vous que je
  les associe ?
 </seg>
</u>
```

```
<u id=u74 who=S6>Oui</u>
<u id=u75 who=C1 >
 <seg id=c1u75seg1 type=sentence n=75>Vous obtiendrez une
  surface en quatre morceaux</seg>
 <seg id=c1u75seg2 type=mf2 n=72>Opération terminée</seg>
</u>
```

Tableau 3-34 - Segmentation des énoncés

Observons le codage du segment <seg id=c1u75seg2 type=mf2 n=72> dans le Tableau 3-34. L'attribut id indique que ce segment a été prononcé par le compère 1, qu'il fait référence à l'énoncé 75 c'est-à-dire à la balise <u> dont l'identifiant est u75. La troisième partie de l'identifiant, seg2, indique qu'il s'agit du deuxième segment de la balise <u>. La valeur de l'attribut type correspond à l'identifiant du message standard « Opération terminée ». L'attribut n quant à lui indique la ligne à laquelle se trouve le segment dans le corpus d'origine.

La transition

L'attribut trans (Tableau 3-35) placé dans la balise <u> a pour but de préciser le type de transition entre deux énoncés comme montré dans le Tableau 3-36.

Attribut	Fonction
trans	indique la nature de la transition les valeurs de trans sont: • pause: l'énoncé commence après une pause. Dans ce cas, sa longueur est caractérisée par l'attribut dur présent dans la balise <pause>. Cet attribut dur peut prendre les valeurs short, médium, long. • latching: l'énoncé commence avec une pause plus courte que la normale. • overlap: il y a superposition de deux énoncés.

Tableau 3-35- Valeurs possibles pour l'attribut trans

Transcription initiale:

```
(S1-T3) :
18 : <who=Compère> Opération effectuée
19 : <who=Sujet> ooo Peut-on tracer...
```

Représentation adoptée:

```
<u id=u18 who=C1>Opération effectué</u>
<u id=u19 who=S1 trans=pause> <pause>Peut-on tracer ... </u>
```

Tableau 3-36 - Exemple de codage d'une pause lors d'une transition entre deux énoncés

Bruitage et morphèmes[24] « semi-lexicaux »

L'utilisateur qui réalise une tâche pour la première fois peut hésiter. Celui-ci peut également émettre des vocalises pour indiquer une confirmation. Ces vocalisations ou hésitations seront mises en évidence par la balise <vocal> (Tableau 3-37).

Transcription initiale:
139 : <who=Compère> Veuillez patienter
140 : <who=Sujet> Hmm, hmm.

Représentation adoptée:
<u id=u139 who=C1>
<seg id c1u139seg1 type=ma2 n=139>Veuillez patienter</u>
<u id=u140 who=S1> <vocal> Hmm, hmm.</vocal></u>

Tableau 3-37 - Exemple de balise vocal

Prononciation incompréhensible

Il peut arriver que la clarté d'un texte oral ne soit pas évidente. La balise <unclear> correspond à un segment non reconnu dans la transcription initiale.

Dénomination d'objets

La balise <name> est introduite pour nommer un objet lors de sa première occurrence (Tableau 3-38).

Transcription initiale:
145: <who=Compère>Les surfaces S1 et S2 ne sont pas visualisées à l'écran

Représentation adoptée:
<u id=u145 who=C1>Les surfaces<name>S1></name>et<name>S2</name>ne sont pas visualisées à l'écran</u>

Tableau 3-38 - Exemple d'utilisation de la balise <name>

[24] Un morphème est l'unité significative minimale. Le morphème lexical est un mot qui peut apparaître de façon autonome. Le morphème semi-lexical est une partie de mot telle qu'un préfixe, un suffixe, etc.

c. Niveau 3

Le niveau 3 complète le niveau précédent d'un point de vue quantitatif et qualitatif.

Il s'agit, d'un point de vue quantitatif, de déterminer la longueur des pauses (Tableau 3-39).

Transcription initiale:
19 : <who=compère> Opération terminée
20 : <who=Sujet> Bien /-/ /-/ créer des surfaces /-/ Ah pourquoi il change de vue?
21 : <who=Compère> Veuillez répéter

Représentation adoptée:
<u id=u19 who=C1>Opération terminée</u>
<u id=u20 who=S4>Bien<pause dur=medium>créer des surfaces
<pause dur=short> Ah pourquoi il change de vue ?
</u>
<u id=u21 who=C1>Veuillez répéter</u>

Tableau 3-39 - Exemple de durée des pauses

Examinons, maintenant, les éléments qualitatifs.

Pour certaines injections, le codeur devra pouvoir indiquer le sens donné à la réponse. D'après les conventions de retranscription du corpus expérimental de GOCAD, « Hmm » est considéré comme un morphème « semi-lexical » indiquant un « bruit » de la part du sujet et analysé comme une confirmation. Ces morphèmes « semi-lexicaux » sont donc décrits dans la balise <vocal> sous l'attribut desc. Les valeurs de cet attribut sont précisées dans le <header> du corpus sous <encodingDesc> (Tableau 3-40). Ainsi, dès que l'on rencontre 'VOC3' dans une balise <vocal>, dans la représentation adoptée, cela signifie qu'une hésitation de type 3 est marquée.

Transcription initiale:
139 : <who=Compère> Veuillez patienter
140 : <who=Sujet> Hmm, hmm.

Représentation adoptée:

```
<header>
<vocal desc='VOC2'> : hésitation de type 2(c'est-à-dire:'Hmm': hésitation)
<vocal desc='VOC3'> : hésitation de type 3(c'est-à-dire: 'Hmm': confirmation)
</header>
...
<u id u139 who C1>
   <seg id c1u139seg1 type=ma2 n=139>Veuillez patienter</u>
<u id=u140 who=S1> <vocal desc='VOC.3' iterated=y> </u>
...
```

Tableau 3-40 - Exemple de représentation du morphème semi-lexical « Hmm »

Dans ce tableau, notons la répétition du « Hmm » qui est indiquée par l'attribut *iterated* (y).

Un autre élément qualitatif est la précision de la raison pour laquelle un segment n'a pas été reconnu. (`<unclear reason='inaudible'>`)

Ce niveau traite également la notion de référence.

Le but du logiciel GOCAD étant de créer des surfaces à partir d'objets de différents types, le sujet fera référence à de nombreux objets qu'il désignera à la fois par des noms mais aussi par des anaphores[25]. La référence ne sera pas limitée aux objets, elle pourra s'appliquer à des actions.

Ce point n'est qu'une brève approche de la référence. Le codage adopté dans cette partie n'est qu'une ébauche de solution. On trouvera en annexe C un document traitant des dernières propositions faites par Florence Bruneseaux en matière de codage de la référence [BRUNE, 96c].

Le logiciel GOCAD permet de créer des surfaces à partir de différents objets tels des lignes ou encore des points. Le sujet fait donc référence à ces objets. La référence n'est pas limitée aux objets, elle s'applique également aux actions.

[25] Un segment de discours est dit anaphorique lorsqu'il est nécessaire, pour donner l'interprétation (même simplement littérale), de se reporter à un autre segment du même discours[DUCROT, 72].

Référence à des objets

Au niveau 2, la balise <name> est utilisée pour nommer un objet lors de sa première occurrence. Au niveau 3, la balise <rs> est introduite pour désigner une chaîne de caractères qui sera utilisée par les occurrences ultérieures d'un objet déjà défini par name (Tableau 3-41).

Transcription initiale:
145 : <who=Compère> Les surfaces S1 et S2 ne sont pas visualisées à l'écran
146 : <who=Sujet> Les visualiser.

Représentation adoptée:

```
<u id=u145 who=C1> Les surfaces
        <name type=objet key=O1>S1</name>
        et <name type=objet key=O2>S2</name>
        ne sont pas visualisées à l'écran
</u>
<u id=u146 who=S7>
        <rs key='O1 O2'>Les</rs> visualiser.
</u>
```

Tableau 3-41 - Exemple de références à des objets

Les deux objets S1 et S2 sont définis dans l'énoncé du compère, ils seront repérés tout au long du dialogue par les clés O1 et O2. Dans la réponse du sujet ces 2 objets sont définis par « les ». Dans la balise <rs>, les deux valeurs de l'attribut key donnent la signification du pronom « les » en désignant explicitement les objets correspondants.

Référence des actions

La référence aux actions diffère de la référence aux objets parce qu'elle se rapporte à un énoncé entier et non à un syntagme[26] défini.

Dans le Tableau 3-42, la balise <ref> de l'élément u id=u34) permet de faire le lien entre l'action référencée (i.e. : *cette opération*) et sa référence (contenue dans le tour de parole u33), à l'aide de l'attribut target.

[26] Un syntagme est un groupement homogène du point de vue de son comportement syntaxique. Ex. syntagme nominal (Il, le chat, Pierre, manger des bananes). verbal (court, aime Marie), prépositionnel (chez Pierre, depuis trois heures etc.)[ROMARY]

Transcription initiale:

```
33 : <who=Sujet> o Tracer l'intersection de H2 et H3 en
      rouge.
34 : <who=Compère> Je ne peux pas faire cette opération
      directement.
```

Représentation adoptée:

```
<u id=u33  who=S2 trans=pause>
     Tracer l'intersection de
          <name type=objet key=O1>H2</name> et
               <name type=objet key=O2>H3</name>en rouge.
</u>
<u id=u34 who=C1> <rs key=S2>Je</rs> ne peux pas faire
     <ref id=r1 target=u33>cette opération</ref>
     directement.
</u>
```

Tableau 3-42 - Exemple de référence à des actions

2. Codage pour le geste

Après avoir traité le codage pour la parole, nous allons effectuer la même démarche pour le geste.

a. Niveau 1

La prise en compte du geste est capitale puisque geste et parole se combinent et se complètent mutuellement. Un dialogue peut être à dominante vocale ou gestuelle en fonction du moyen d'expression choisi par l'utilisateur. Suivant le type de dialogue, le geste complète la parole ou inversement. Certaines catégories de gestes directement liées à l'application, comme l'utilisation de la souris, sont nécessaires afin de faire apparaître l'ensemble des informations communiquées entre le sujet et le compère.

A ce niveau, le but n'est pas d'identifier le moment précis où le geste a lieu. Il s'agira plutôt d'indiquer si le geste a lieu dans le discours à un moment relatif par rapport à la parole ou s'il y a synchronisation entre discours et geste.

La sélection ou désignation d'un objet

Les gestes pris en compte sont ceux correspondant à une utilisation de la souris pour désigner un objet sur l'écran ou le sélectionner en cliquant dessus. Dans le codage adopté, une ancre (balise <anchor>) est insérée dans le dialogue au moment du geste. La

balise <kinesic> permet d'identifier l'auteur du geste à l'aide de l'attribut who et décrit le geste à l'aide de l'attribut desc (Tableau 3-43).

Transcription initiale:
50 : <who= Sujet> Peut-on éliminer de la surface H la surface que l'on vient, la surface comprise entre la ligne euh, <CLICKO # courbe d'intersection> des points d'intersection que je désigne là ?

Représentation adoptée:
<u id=u50 who= S1>Peut-on éliminer de la surface H2, la surface que l'on vient, la surface comprise entre la ligne euh, <anchor id=P1> des points que je désigne là <anchor id=P2> ?
</u>
<kinesic desc="click sur la courbe d'intersection" start=P1 end=P2>

Tableau 3-43 - Exemple de sélection ou de désignation d'un objet

Le déplacement de la souris

Dans le corpus initial, un déplacement de la souris est indiqué par la balise <CLICKDEP>. Elle signifie que le sujet décrit un mouvement assez long avec la souris en gardant le bouton enfoncé. L'élément <kinesic> sera également utilisé pour décrire un déplacement de la souris. Cet élément pourra être complété par l'attribut iterated indiquant si le geste décrit est répété ou unique.

L'attribut synchro de la balise <anchor> indique s'il y a synchronisation entre le geste et la parole. Cette description sera complétée par un élément <xptr> qui indique les caractéristiques précises du déplacement.

Dans cette description du déplacement de la souris à l'écran, les attributs de <xptr> auront plus précisément les significations reprises dans le Tableau 3-44.

Attribut	fonction
doc	spécifie le document correspondant au déplacement décrit
from(xy)	localisation sur l'écran du début du mouvement
to(xy)	localisation sur l'écran de la fin du mouvement
id	identifiant de l'élément décrit

Tableau 3-44 - Signification des attributs de la balise <xptr>

Transcription initiale:
70 : <who=Sujet> <CLICKDEP # surface> Augmentez la densité du maillage autour de ces points

Représentation adoptée:
<u id=u70 who=S8> <anchor id=a1 synchro=k1>Augmentez la densité du maillage autour de ces points</u>
<kinesic id=k1 desc="désignation d'une surface par un mouvement long iterated=n>
<xptr id=xptr1 doc=maillage from='space(xy)' to='space(x'y')' target=k1>

Tableau 3-45 - Exemple de notation du déplacement de la souris

Déplacement d'objets

La description d'un déplacement d'un objet sera similaire à celle d'un déplacement de la souris. A la différence du cas précédent, la localisation initiale définie pas `from(x,y)` ne désignera pas seulement un lieu à l'écran mais aussi l'objet à déplacer.

b. Niveau 2

Le niveau 2 complète le niveau 1 à l'aide d'attributs et de balises qui fournissent une description plus approfondie du geste. Ce niveau propose notamment d'indiquer lorsque c'est possible l'instant précis où le geste a lieu.

Ce niveau apporte une précision temporelle par l'ajout de la balise <when> (Tableau 3-46).

Transcription initiale:
81: <who = Sujet> Supprimez <CLICKO# sur la coupe> la partie désignée par la souris

Représentation adoptée:
<when id=u81 absolute="12:20:01:01">
<u id=u81 who=Sujet> Supprimez <anchor id=a1> la partie désignée par la souris
<kinesic desc="click sur la coupe" target=a1>
<when id=k1 interval=15 since=u81>

Tableau 3-46 - Exemple d'utilisation de la balise <when>

La première ligne donne le moment auquel débute le tour de parole u81. Les deux lignes suivantes décrivent le geste effectué avec la souris. La dernière ligne, quant à elle,

précise que le geste a lieu lors du tour de parole u81, quinze unités de temps après son commencement.

Le geste est décrit par des indications particulières telles que la forme du déplacement entre la position initiale et la position finale de l'objet, le sens du déplacement, la vitesse de déplacement.

Le niveau 2 propose, également, d'élargir la balise <shift> au niveau du geste. Au départ, cette balise est définie par la TEI pour mettre en évidence des changements significatifs au niveau du langage dans un énoncé. L'élargissement de l'application de la balise <shift> au geste permettrait de faire apparaître un changement dans une caractéristique du déplacement. Les attributs nécessaires pour réaliser un tel codage sont décrits dans le Tableau 3-47.

Id	identifiant
feature	désigne le sujet de la modification :la forme, le sens et la vitesse.
New	contient la nouvelle valeur après le changement
target	réalise le lien entre la balise <shift> et <xptr>

Tableau 3-47: Attributs de la balise <shift>

Une accélération du geste pourra être codée comme dans le Tableau 3-48.

```
<xptr id=xptr1 doc="maillage" from='space (xy) to='space (x',y')'>
<shift id=s1 feature=vitesse new=aa target=xptr1>
```

Tableau 3-48: Exemple de codage d'accélération d'un geste

c. Niveau 3

Le niveau 3 est très peu développé. Il pourrait être utilisé si le niveau 2 ne décrivait pas suffisamment le geste. Un mouvement ou un déplacement serait décomposé plus finement en une suite de positions.

3. L'utilisateur et l'environnement dans lequel la tâche évolue

a. Action du sujet sur l'environnement

Le sujet peut faire agir le compère :

- directement sur la tâche ;
- sur l'environnement.

Le Tableau 3-49 caractérise le cas où le sujet veut faire agir le compère directement sur la tâche.

Transcription initiale:

```
75 : <who = Sujet> Supprimer la partie de /-/ H2 en dessous
      de H3
```

Représentation adoptée:

```
<u id=u75 who=S8>Supprimer la partie de<pause dur=short>
     <name type=object key=O1>H2</name>en dessous de
     <name type=object key=O2>H3</name>
</u>
```

Tableau 3-49 - Action du compère directement sur la tâche

Lorsque le sujet veut agir sur l'environnement, il ne cherche pas à modifier l'objet qui est à l'écran mais désire le visualiser différemment. Cette visualisation pourra se faire à l'aide de l'outil pilote de caméra. Le codage adopté pour les déplacements des acteurs dans une pièce de théâtre a été adapté aux mouvements du pilote de caméra (utilisation de la balise move).

Afin de donner un maximum d'informations, qu'il s'agisse du déplacement d'objets (le point de départ du mouvement, le type de déplacement, le sens du déplacement) ou du pilote de caméra (le sens et la limite du déplacement), il s'agit de compléter les indications données par la TEI.

Les informations nécessaires au déplacement d'un objet se trouvent dans les balises <kinesic> et <move>.

Concernant l'objet particulier *pilote de caméra*, l'idée est d'adapter la balise <move> à ses mouvements spécifiques. L'attribut who indiquera la caméra qui subit l'action, le type de déplacement sera précisé dans *type* et where indiquera le point d'arrivée du mouvement, la limite du déplacement devra être directement recherchée dans l'énoncé du sujet et sera souvent exprimée en degrés.

Transcription initiale:
99 : <who = Sujet> o Faites pivoter la caméra de 90 degrés vers le haut
100 : <who = Compère> Bien compris
...
104 : <who = Compère> Opération terminée

Représentation adoptée:

```
<u id=u99 who=S8 trans=pause><pause=short>Faites pivoter la caméra de
    <num type=degree id=limite1 value='90'>90 degrés</num> vers le haut</u>
<u id=u100 who=C1>
   <seg id=c1u100seg1type=mc2 n=100>Bien compris</seg>
   <seg id=c1u100seg5 type=sentence copy0f=u69c1seg2 n=104> Opération
terminée</seg>
</u>
<move type=pivoter who=caméra1 id=mpc1 where=limite1>
```

Tableau 3-50 - Action du compère sur l'environnement

La limite indiquée par la valeur "limite1" dans <move> fait référence à la limite du mouvement défini dans l'énoncé u (id=99) c'est-à-dire "90 degrés". II nous restera à indiquer dans ce codage que le sens est ici "vers le haut".

b. Perception de la tâche par le sujet

Cette partie a pour objet de fournir des informations techniques afin de décrire l'objet tel qu'il se présente au sujet. Il s'agit de décrire le résultat d'une action sur la perception de l'objet.

La balise <tech> va permettre à la fois de décrire une caractéristique technique de l'objet lui-même (ex. sa couleur, sa forme,...) et la façon dont le sujet perçoit l'objet à l'écran. Elle décrit une caractéristique technique de l'objet .

La balise <tech> dispose des attributs type et what. L'attribut type définit la modification à apporter et peut prendre les valeur light, color, form, view. L'attribut what indique l'objet décrit. La balise <tech> est toujours en relation avec une balise <shift>.

La balise <shift> permet de décrire un changement d'une caractéristique technique. Le trait modifié est indiqué sous feature et la nouvelle valeur sous new.

Dans le Tableau 3-51, la balise <tech> (id = tq1) caractérise le type de modification (attribut type) à apporter à l'objet H3 (attribut what). La balise <shift>

indique que la nouvelle couleur (attribut `feature`) pour l'objet décrit dans la balise `<tech>` identifiée par `tq1` (attribut `target`) est le blanc (attribut `new`).

Transcription initiale:

```
110 : <who = sujet> Faites apparaître la surface H3 en blanc
111 : <who = Compère> Bien compris
112 : <who = Compère> Opération en cours
113 : <who = Compère> Opération en cours
114 : <who = Compère> Veuillez patienter
115 : <who = Compère> Opération en cours
116 : <who = Compère> Opération terminée
```

Représentation adoptée:

```
<u id=u110 who=S8>
   Faites apparaître la surface <rs key=02>H3</rs> en blanc </u>
<u id=u111 who=C1>
      <seg id=c1u111seg2 type=mc2  n=111>Bien compris</seg>
      <seg id=c1u111seg3 type=ma3  n=112>Opération en cours</seg>
      <seg id=c1u111seg4 type=ma3  n=113> Opération en cours</seg>
      <seg id=c1u111seg5 type=ma2  n=114>Veuillez patienter</seg>
      <seg id=c1u111seg6 type=ma3  n=115>Opération en cours</seg>
      <seg id=c1u111seg7 type=mf2  n=116>Opération terminée</seg>
</u>
<tech type=color what=H3 id=tq1>
<shift id=s1  feature=color  new=blanc  target=tq1>
```

Tableau 3-51 - Changement d'une caractéristique technique

L'éditeur de dialogue aura comme objectif de faciliter les diverses manipulations qui peuvent être effectuées sur les dialogues tels que ceux du corpus GOCAD et plus généralement sur tout document SGML.

3.2 Aide à l'analyse grammaticale de texte

Si, de plus en plus, lors de l'utilisation de systèmes informatiques, on désire une interface multimodale[27], il est nécessaire, non seulement, de permettre d'inclure les différents modes de communication dans un même document, mais aussi de comprendre le discours de l'utilisateur, qu'il soit écrit ou parlé.

[27] Interface permettant l'utilisation de plusieurs modes de communications simultanés comme le clavier, la désignation par la souris mais aussi le geste et la parole.

Une fois le discours transcrit en une forme manipulable, la machine devra l'analyser pour l'interpréter. Cette interprétation est nécessaire si l'on veut automatiser le traitement de documents et particulièrement si l'on veut qu'elle comprenne les ordres d'un opérateur humain.

Cette analyse se décompose en une partie syntaxique et une partie sémantique. De nombreuses discussions [LOPEZ, 96] ont actuellement lieu sur le point de savoir à quel niveau doit se faire la séparation entre ces deux analyses. Mais, de plus en plus, la tendance des recherches s'oriente vers un « gonflage » du lexique au détriment de l'intelligence placée dans l'analyseur syntaxique ou sémantique.

Si "*la syntaxe traite de la combinaison des mots dans une phrase*" [DUCROT, 72], il faut décrire l'ensemble de règles régissant ces combinaisons. Les types de grammaires sont nombreux. Parmi ceux-ci, citons la Grammaire d'arbres adjoints [JOSHI, 75] qui est utilisée au C.R.I.N. Notons, également que cette notion d'arbre adjoint permet d'intégrer les différents niveaux de l'analyse et d'y intégrer les notions de lexique, de syntaxe et de sémantique [LOPEZ, 96].

L'objectif de l'application demandée est donc de permettre la visualisation de grammaire d'arbres adjoints à l'aide d'un éditeur graphique d'arbres SGML. En effet, une manière de représenter un document SGML est de le faire sous la forme d'un arbre. On pourra vérifier la correspondance entre l'exemple de document SGML, contenu dans le Tableau 3-52, et sa représentation sous forme d'arbre présentée dans la Figure 2 page 70.

```
<p>
<cl type='finite declarative' function='independent'>
  <phr type=NP function='subject'>Nineteen fifty-four,
   <cl type='finite relative declarative'
    function='appositive'>when
    <phr type=NP function='subject'>I</phr>
    <phr type=VP function='predicate'>was eighteen
              years old</phr>
   </cl>,
  </phr>
  <phr type=VP function='predicate'>
   <phr type=V function='main verb'>is held</phr>
   <phr type=NP function='complement'> <!-- ? -->
    <cl type='nonfinite' function='predicate nom.'>
     <phr type=V function='copula'>to be</phr>
     <phr type=NP function='predicate nom.'>
      a crucial turning point
      <phr type=PP function='postmodifíer'>in
```

```
        <phr type=NP function='prep.obj.'>
         the history
         <phr type=PP function='postmodifier'>
          of the Afro-American</phr>
        </phr>
       </phr>
       ---
       <phr type=PP
         function='appositive postmodifier'>for
        <phr type=NP function='prep.obj.'>the U.S.A.
         <phr type=PP function='postmodifier'>
          as a whole</phr>
           </phr>
          </phr>
      </phr>
      ---
      <phr type=NP'function='appositive pred.nom.'>
       the year
          <cl type='finite relative'
            function='adjectival'>
           <phr type=NP function='subject'>
            segragation</phr>
        <phr type=VP function='predicate'>
         <phr type=V function='main verb'>
             was outlawed</phr>
         <phr type=PP function='postmodifier'>
             by the U.S. Supreme Court</phr>
        </phr>
       </cl>
      </phr>
     </cl>
    </phr>
   </phr>
.
</cl>
<cl type='finite declarative' function='independent'>
 <phr type=NP function='subject'>It</phr>
 <phr type=VP function='predicate'>
  <phr type=V function='main verb'>was</phr>
  also
  <phr type=NP function='prédicate nom.'>
   a crucial year for me</phr>
 </phr>
 <cl type='finite declarative'
  function='dependent causative'>because
```

```
 <phr type=PP function='sentence adverb'>
  on June 18, 1954</phr>,
 <phr type=NP function='subject'>I</phr>
 <phr type=VP function='predicate'>
  <phr type=V function='main verb'>began serving</phr>
  <phr type=NP function='complement'>
   a sentence in state prison
   <phr type=PP function='complement'>
    for possession of marijuana
   </phr>
  </phr>
 </phr>
</cl>
</cl>
.
</p>
```

Tableau 3-52 - Fichier SGML

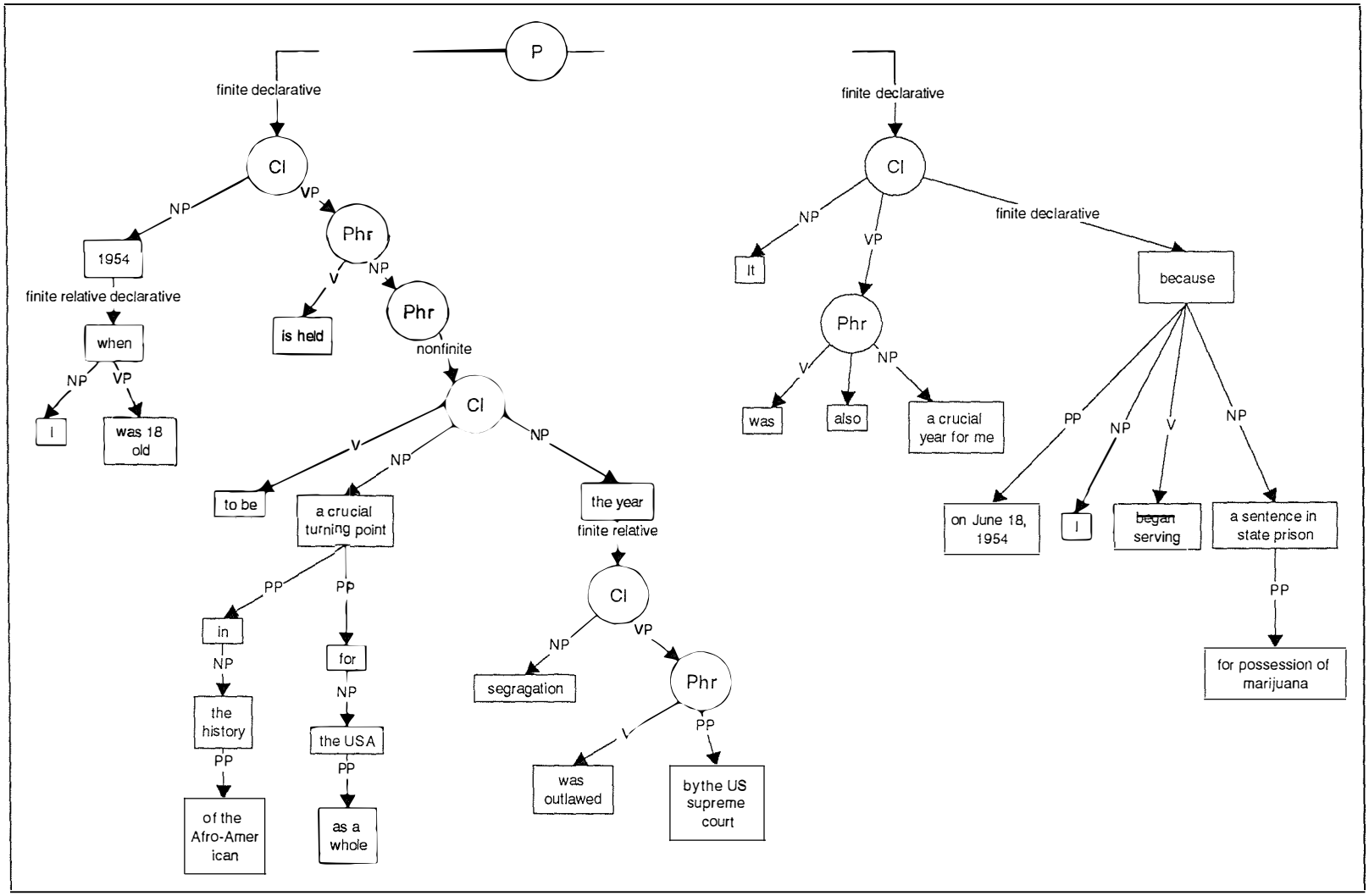

Figure 2 - Exemple de document SGML représenté sous forme d'arbre

3.2.1 Arbres Adjoints

Dans ce point, nous décrivons d'une manière intuitive la notion d'arbre adjoint ; ce qui nous permettra de bien comprendre à quoi sert une application qui manipule des arbres adjoints..

Ces arbres permettent, d'abord, la description passive d'une grammaire. Ensuite, en leur adjoignant la notion de structures de traits[28], ils seront utilisés dans le processus d'analyse de texte proprement dit. En effet, il est possible de vérifier la correspondance d'une phrase à sa grammaire en combinant des unifications et des dérivations (adjonctions[29] et/ou substitutions[30]) des arbres relatifs à la phrase à analyser.

Les arbres adjoints sont séparés en deux catégories : ils sont tantôt élémentaires, tantôt dérivés.

L'arbre élémentaire représente les morphèmes. Il est de profondeur quelconque. Il a la particularité de posséder une (ou plusieurs) feuille(s) occupée(s) par un mot terminal appelé *ancre ou tête (co-têtes ou têtes multiples)*. Cette ancre, lors de la phase d'analyse, permet de ne sélectionner que les arbres qui ont une tête présente dans la phrase à analyser. Ainsi, lors de l'analyse de la phrase "le chat boit", on ne prendra en compte que les arbres contenant les ancres boit, chat et le.

Un arbre élémentaire est initial ou auxiliaire. L'arbre initial est composé de nœuds représentant des unités grammaticales. Ils sont soit des mots terminaux, soit des lemmes non terminaux où se feront les substitutions ultérieures (Figure 3).

La particularité des arbres auxiliaires, par rapport aux arbres initiaux, est que leurs feuilles sont de la même catégorie que celle de leur nœud racine (Figure 4). Ils permettent de représenter des modificateurs (adjectifs, adverbes, relatives), les verbes à complétives, des verbes modaux et des auxiliaires.

---

[28] Les structures de traits forment un mécanisme propre aux grammaires d'unification. Ces traits sont un ensemble de paires attributs - valeurs décrivant une unité lexicale. Ces structures permettent, lors d'une unification, de vérifier la cohérence entre les deux parties unifiées. Imaginons que nous sommes en train d'unifier un nom avec son déterminant ; les structures de traits associées à chacun d'eux permettront de vérifier qu'ils sont de même genre et de même nombre (voir [LOPEZ, 96]).

[29] L'adjonction permet d'insérer un arbre auxiliaire à un nœud quelconque de même famille dans un autre arbre.

[30] La substitution correspond à l'opération de réécriture de grammaire hors contexte.

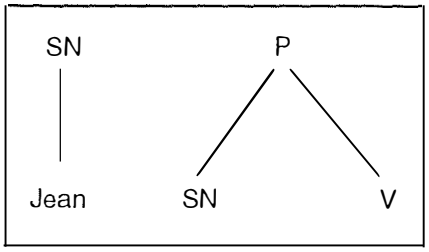

Figure 3 - Exemples d'arbres adjoints initiaux

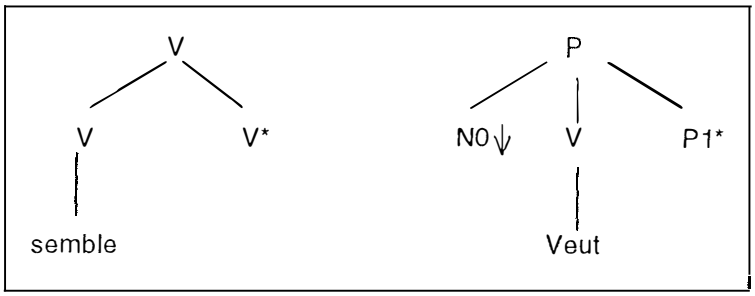

Figure 4 - Exemples d'arbres adjoints auxiliaires

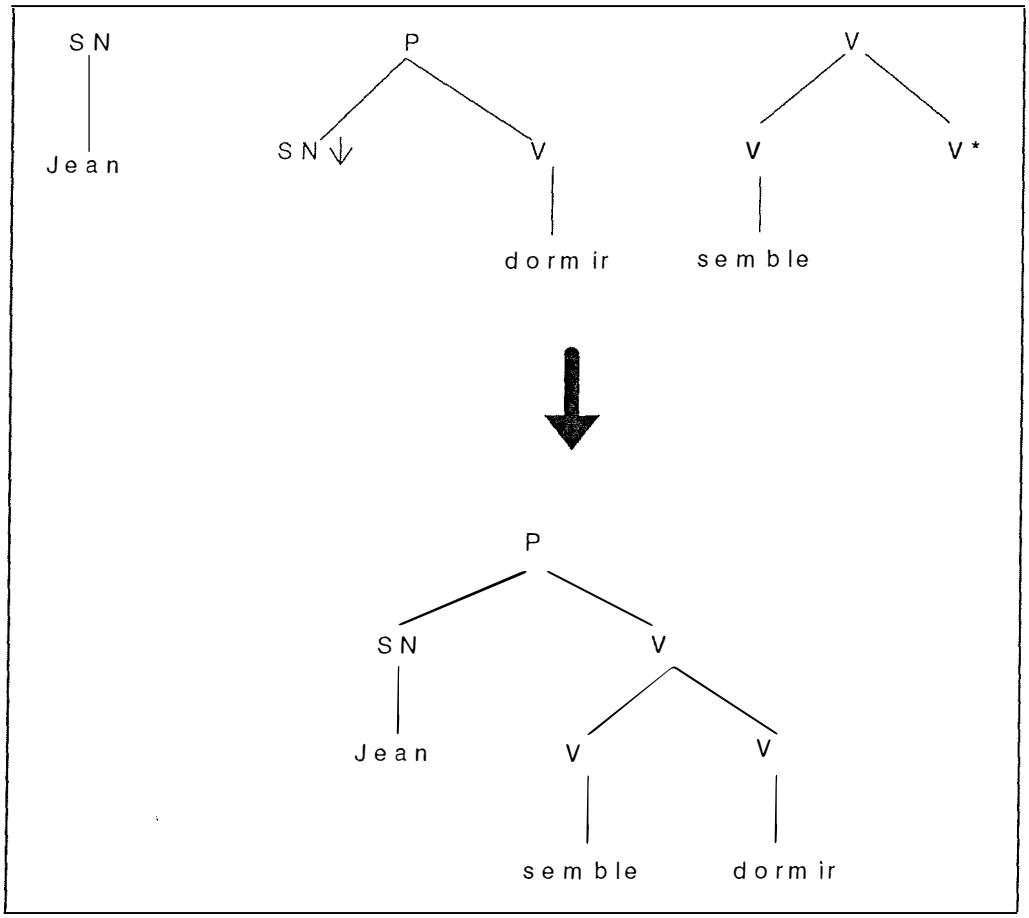

Figure 5 - Exemple d'arbre adjoint dérivé

Un arbre dérivé (Figure 5) est un arbre obtenu par combinaison d'arbres élémentaires par des procédures de dérivation (substitution et adjonction).

3.2.2 Arbres Adjoints et SGML

Nous avons vu, dans le Tableau 3-52 et la Figure 2, qu'un document SGML peut être vu comme une arborescence de balises. On pourrait donc imaginer de représenter les arbres adjoints à l'aide de SGML et créer une DTD les décrivant. Ce qui nous permettrait d'utiliser un éditeur de documents SGML sous forme d'arbres pour visualiser les arbres adjoints.

Cette DTD ne peut être une DTD standard comme la TEI. Une description faite à l'aide d'une DTD standardisée pose problème. En effet, il s'agit de décrire une grammaire, et ici plus particulièrement un arbre, à l'aide d'une norme faite pour l'encodage de texte. Il n'existera donc pas de balises spécifiques à la structuration d'arbres, même si un document SGML est structuré sous la forme d'un arbre.

Les attributs spécifiques aux arbres adjoints ne sont pas représentables à l'aide des attributs disponibles dans la TEI. L'utilisation de ces derniers obligerait non seulement à détourner la signification normale des balises, mais elle rendrait également la lecture des documents résultants très lourde, voire impossible.

En effet, la TEI fournit un certain nombre de mécanismes permettant de rassembler différentes balises, ainsi que les lier entre elles[32]. Malheureusement, le codage qui en résulte ne rend pas le document directement lisible. L'adaptation du logiciel de visualisation susmentionné à l'analyse d'un tel document ne correspond pas à la volonté de le rendre indépendant de la DTD utilisée pour un document. Elle exigerait également une programmation beaucoup trop spécifique. Nous devrions prévoir une méthode qui transforme le résultat de l'interprétation du document TEI en vue de son affichage et de sa manipulation, au sein même de l'éditeur. De plus, l'élaboration et les éventuelles modifications d'une DTD sont de loin moins coûteuses en temps et en investissement que la programmation, même si cette dernière se fait à l'aide d'outils tels que LEX ou YACC.

Nous allons donc profiter de la puissance offerte par les DTD et voir comment nous pouvons les utiliser pour visualiser un arbre adjoint sans perdre de sémantique.

[32] A l'aide de balise telle que : <link>, <f>, <fs>, <fslib> et .

3.2.3 Une DTD pour les arbres adjoints.

Idéalement, nous devrions essayer de placer dans les balises les informations relatives aux nœuds des arbres adjoints (nom, type, structure de traits). Si nous voulons voir apparaître le nom du nœud (par exemple, groupe verbal, groupe nominal, etc.), nous devons déterminer une DTD où le "*tag*" d'une balise serait ce nom. C'est malheureusement difficilement concevable, cela nécessiterait la construction d'une DTD au même moment que la grammaire elle-même. Ceci ne correspond pas à la philosophie de SGML.

Il est extrêmement important de pouvoir visualiser de façon très simple un arbre adjoint grâce à une représentation telle que dans la Figure 6.

S'il est nécessaire de construire simultanément la DTD et la grammaire, il est préférable de décréter que cette dernière est superflue.

Nous allons imaginer une DTD dite *interne*. Etant inexistante, elle ne permettra pas de vérifier, à l'aide d'un `Parser` de document SGML, la validité d'un élément (d'un arbre) de la grammaire vis-à-vis de cette DTD. Cette DTD nous rappellera de prêter attention au respect des conventions habituellement liées aux documents SGML. Cette représentation interne des arbres adjoints a comme principe de :

- remplacer le tag des balises par le nom du nœud de l'arbre adjoint ;
- placer les attributs du nœud (*top*, *bottom*[33]) dans la balise.

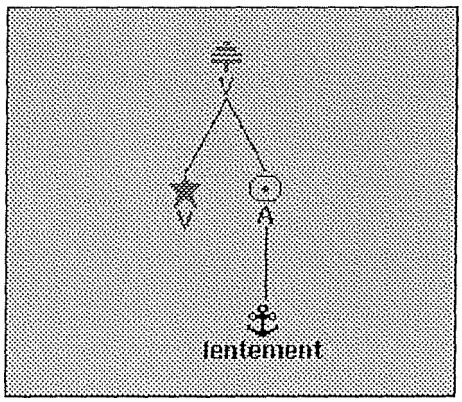

Figure 6 - Exemple de représentation d'un arbre adjoint

[33] Correspondant aux traits de l'unité grammaticale contenue dans le nœud, voir 28. Chaque trait (*bottom* ou *top*) contient les informations de genre et nombre de l'unité lexicale.

Malgré que nous n'ayons défini aucune DTD décrivant ce document, nous utiliserons un format qui se rapproche très fort de la structure d'un document SGML. L'important est de pouvoir manipuler un arbre adjoint à l'aide d'un éditeur général d'arbres SGML. A cet effet, on codera, sous une forme SGML, l'arbre adjoint de la Figure 6 tel que dans le Tableau 3-53.

```
<V type="0" NA="1" bottom="" >
        <V type="3" top=" ">
        </V>
                <A type="1" top=" " bottom=" " >
                        <lentement type="2">
                        </lentement>
                </A>
</V>
```

Tableau 3-53 - Code correspondant à une DTD interne d'un arbre adjoint

3.2.4 Utilisation de la TEI pour coder un arbre adjoint

Si nous utilisons les balises offertes[34] par la TEI pour coder l'arbre de la Figure 7, nous aboutirons au code présenté dans le Tableau 3-59. Cette figure ne comprend pas un ensemble de traits cohérents, il s'agit d'un exemple reprenant la plupart des type de traits existants.

Pour comprendre un tel code, il est nécessaire d'expliquer les conventions que nous avons dû prendre pour :

- représenter un noeud ;
- représenter le lien de filiation entre deux noeuds ;
- représenter les attributs d'un noeud et leur valeur.

[34] Voir Annexe A et B.

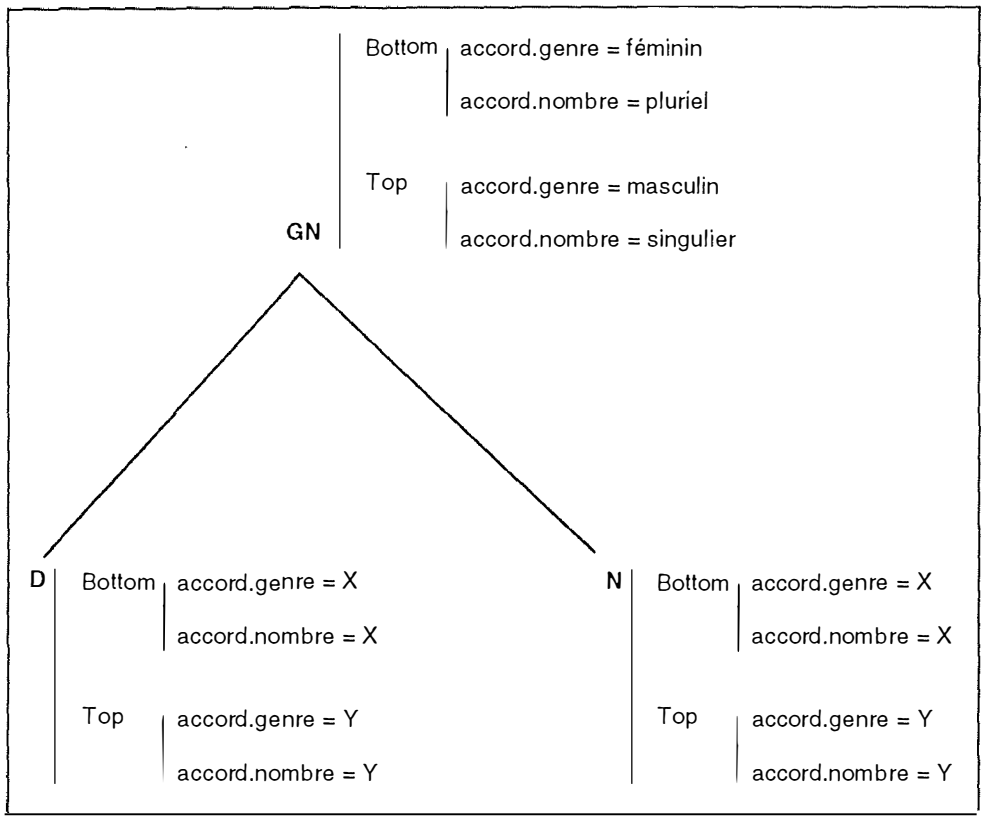

Figure 7 - Exemple d'arbres adjoints et traits associés

A) Représentation d'un nœud

La représentation du nom d'une feuille de l'arbre adjoint se fait à l'aide de l'attribut value de l'élément . Le nœud N de l'arbre adjoint de la Figure 7 est donc représenté par <span id = "spGN1" value "GN" teiform = "span">. On rassemblera tous les éléments de ce type dans l'élément <spangrp teiform = "spangrp">.

La nature[35] du nœud (étoile, flèche, racine, interne) se traduit à l'aide de l'attribut type de l'élément join ou span.

B) Représentation d'un lien père–fils

On représentera la relation de filiation entre deux nœuds en utilisant l'élément join. Ainsi, la racine de l'arbre de la Figure 7 est représentée par <join id= "jGN1"

[35] Flèche, étoile ou ancre.

target = "ID(spD1) ID(spn1) " scoop = "root" targorder= "Y" teiform = "join" >. La racine de l'arbre est qualifiée par l'attribut scoop = "root".

C) Représentation des attributs et de leurs valeurs

La balise <link> de la TEI va nous permettre d'associer aux nœuds leurs traits. Cet élément est compris dans l'élément <linkgrp targorder = "U" teiform = "linkGrp"> (Tableau 3-54).

Le premier mot de la valeur de l'attribut target identifiera le nœud et le second désignera les traits de l'élément N de la Figure 7.

```
<linkgrp targorder = "U" teiform = "linkGrp" >
   <link targorder= "U" targets= "ID(spN1) ID(N1tb) " teiform= "link">
</linkgrp>
```

Tableau 3-54 – Lien entre le nœud et ses traits

Ces traits sont repris dans un élément <fs type = "TB"> qui permettra de désigner la paire de traits associée au nœud (Tableau 3-55). Cet élément <fs> est compris dans l'élément <fslib type = "Top Bottom" teiform = "fsLib">.

```
<fslib type = "Top Bottom" teiform = "fsLib">
   <fs id= "N1tb" type = "TB" feats= "ID(cmf-X) ID(cmf-Y) " rel= "sb" teiform= "fs">
</fs>
</fslib>
```

Tableau 3-55 – Désignation de la paire de traits

L'attribut feats nomme les identifiants des éléments <fs type = "cmf"> permettant de désigner les attributs des noeuds de l'arbre adjoint. L'élément <fs>, à l'aide de l'attribut feats, désigne les attributs de l'unité lexicale représentée (genre et nombre). Les éléments <fs> sont rassemblés dans l'élément <fslib type = "composed morphological feature" teiform = "fsLib"> (Tableau 3-56).

```
<fslib type = "composed morphological feature" teiform = "fsLib">
   <fs id = "cmf-X" type = "cmf" feats = "ID(mf-gx) ID(mf-ny)" rel = "sb" teiform =
"fs"> </fs>
   <fs id = "cmf-Y" type = "cmf" feats = "ID(mf-gx) ID(mf-ny) " rel = "sb" teiform =
"fs"> </fs>
</fslib>
```

Tableau 3-56 – Attributs genre et nombre de l'unité lexicale N

Les valeurs morphologiques sont rassemblées dans l'élément <fs rel = "sb" type = "mf" teiform = "fs"> compris dans l'élément <fslib type = "morphological feature values" teiform= "fsLib">. Les valeurs morphologiques sont du type : "accord.genre = X". Elles sont reprises dans des éléments <f> dont l'attribut name donne la valeur gauche de l'équation et l'attribut fval la valeur droite. Le terme droit peut être une variable ou une constante (Tableau 3-57).

```
<fslib type "morphological feature values" teiform "fsLib">
   <fs rel = "sb" type = "mf" teiform "fs">
        <f id= "mf-gX" name = "accord.genre" rel = "eq" fval = "ID(varX) "   teiform
= "f">
        </f>
   </fs>
<fslib>
```

Tableau 3-57 – Valeur de l'attribut est une variable

L'attribut fval pointe donc tantôt à une variable tantôt à une constante (Tableau 3-58)

```
constante
<f id= "ag-f" name = "féminin" rel= "eq" teiform= "f"> </f>
variable
<f id= "varX" name = "X" rel= "eq" teiform= "f"> </f>
```

Tableau 3-58 - Type de valeurs de l'attribut fval

L'utilisation de variables permet, lors de l'unification de deux termes lexicaux, de vérifier qu'il pointent bien vers la même valeur, ce qui serait impossible avec des constantes. En effet, le hasard pourrait attribuer une valeur identique aux deux attributs que l'on cherche à unifier.

```
<tei.2 teiform = "TEI.2">
<teiheader type = "text" status = "new" teiform = "teiheader">
...
</teiheader>
<text teiform = "text">
<fslib type "morphological feature values" teiform "fsLib">
   <fs rel = "sb" type = "mf" teiform "fs">
        <f id= "mf-gX" name = "accord.genre" rel = "eq" fval = "ID(varX) " teiform =
"f">
        </f>
        <f id= "mf-nY" name = "accord.nombre" rel = "eq" fval = "ID(varY)" teiform
= "f">
        <f id= "mf-ns" name = "accord.nombre" rel = "eq" fval = "ID(ag-s) "teiform =
"f">
        <f id= "mf-np" name = "accord.nombre" rel = "eq" fval = "ID(ag-p) " teiform
= "f">
        <f id= "mf-nf" name = "accord.nombre" rel = "eq" fval = "ID(ag-f) " teiform =
"f">
        <f id= "mf-nm" name = "accord.nombre" rel = "eq" fval = "ID(ag-m) " teiform
= "f">
        </f>
<f id= "varX" name = "X" rel= "eq" teiform= "f"> </f>
<f id= "varY" name = "Y" rel= "eq" teiform= "f"> </f>
<f id= "ag-f" name = "feminin" rel= "eq" teiform= "f"> </f>
<f id= "ag-m" name = "masculin" rel= "eq" teiform= "f"> </f>
<f id= "ag-p" name = "pluriel" rel= "eq" teiform= "f"> </f>
<f id= "ag-s" name = "singulier" rel= "eq" teiform= "f"> </f>
   </fs>
<fslib>
<fslib type = "composed morphological feature" teiform = "fsLib">
   <fs id = "cmf-X" type = "cmf" feats = "ID(mf-gx) ID(mf-ny) " rel = "sb" teiform =
"fs"> </fs>
   <fs id = "cmf-Y" type = "cmf" feats = "ID(mf-gx) ID(mf-ny) " rel = "sb" teiform =
"fs"> </fs>
<fs id = "cmf-ms" type = "cmf" feats = "ID(mf-gm) ID(mf-ns) " rel = "sb" teiform =
"fs"> </fs>
<fs id = "cmf-fp" type = "cmf" feats = "ID(mf-gf) ID(mf-np) " rel = "sb" teiform =
"fs"> </fs>

</fslib>

<fslib type = "Top Bottom" teiform = "fsLib">
<fs id= "GN1tb" type = "TB" feats= "ID(cmf-ms) ID(cmf-fp) " rel= "sb" teiform=
"fs"> </fs>
```

```
    <fs id= "N1tb" type = "TB" feats= "ID(cmf-X) ID(cmf-Y)" rel= "sb" teiform= "fs">
</fs>
<fs id= "D1tb" type = "TB" feats= "ID(cmf-X) ID(cmf-Y) " rel= "sb" teiform= "fs">
</fs>

</fslib>

<spangrp teiform = "spanGrp">
<join id= "jGN1 target = "ID(spD1) ID(spN1) " scoop = "root" targorder= "Y"
teiform = "join" >
<span id= "spD1" value = "D" teiform = "span">
<span id= "spN1" value = "N" teiform = "span">
</spangrp>

<linkgrp targorder = "U" teiform = "linkGrp" >
<link targorder= "U" targets= "ID(spGN1) ID(GN1tb)" teiform= "link">
<link targorder= "U" targets= "ID(spD1) ID(D1tb)" teiform= "link">
<link targorder= "U" targets= "ID(spN1) ID(N1tb)" teiform= "link">
</linkgrp>

</text>
</tei.2>
```

Tableau 3-59 - Code TEI complet représentant un arbre adjoint

3.3 Conclusion

Dans ce chapitre, nous nous sommes familiarisés, dans un premier temps, avec les spécificités liées à la manipulation des corpus de dialogues, et plus particulièrement celles liées au corpus de dialogue GOCAD. Dans un second temps, nous avons abordé les particularités des arbres adjoints.

Nous allons pouvoir, dans le chapitre suivant, dégager les fonctionnalités qui doivent faire partie des outils les manipulant.

4. Fonctionnalités des éditeurs SGML

Nous verrons dans ce chapitre les fonctionnalités se rapportant aux deux types d'application. Ces applications sont celles présentées au chapitre précédent : l'aide à l'édition tant de dialogues que d'arbres SGML. A priori, elles sont dissemblables, mais en réalité, elles différent seulement au niveau de leur interface.

Nous allons dans ce chapitre décrire les différentes fonctionnalités attendues de ces applications. Ensuite, nous essayerons de voir s'il en existe d'autres, liées à des applications de même type, et si nous pouvons les rassembler. Nous terminerons en analysant le paradigme soulevé par de telles fonctionnalités.

Nous avons groupé les fonctionnalités en trois grandes catégories qui sont le filtrage, la manipulation des éléments contenus dans un document et l'exportation des documents.

Le filtrage reprend la visualisation du document qui doit permettre à l'utilisateur de spécifier le format sous lequel il veut manipuler son document.

La manipulation permet non seulement la modification des éléments du document, mais aussi l'accès aux données proprement dites (modifications du document "physique"). Toutes les fonctionnalités modifiant un document posent un réel problème de cohérence. En effet, comment assurer le respect de la DTD lors de l'ajout et l'effacement d'éléments et même d'attributs ? Sont-ils autorisés à la place désignée ? Les attributs requis sont-ils présents ? Ont-ils une valeur admise ?

Il est nécessaire de concevoir une fonctionnalité commune qui permet l'exportation de documents. En effet, les formats sous lesquels sont manipulés les corpus de dialogues sont nombreux et différents d'un groupe de travail à un autre. Dès lors, il est intéressant de permettre le partage des dialogues traités. Une telle transformation serait également utile pour les arbres adjoints. Dans un premier temps, cet outil doit pouvoir transformer un document TEI en document HTML en vue de sa présentation sur le Web ou, encore, permettre des conversions de/vers Latex. Il en va de même pour les arbres adjoints.

Nous ne détaillerons pas l'exportation dans les paragraphes qui décrivent les fonctionnalités des deux éditeurs. En effet, l'unique fonctionnalité, qui est de convertir un format dans un autre, vient d'être expliquée.

4.1 Fonctionnalités pour l'éditeur de dialogues

Rassemblons les fonctionnalités que nous avons pu dégager lors de rencontres avec les utilisateurs de corpus de dialogues.

4.1.1 Filtrage

Affichage des balises

L'utilisateur, suivant le type de travail qu'il doit effectuer, souhaite masquer les balises dont il n'a pas l'utilité. Il doit, donc, pouvoir choisir les balises qu'il veut afficher ou masquer.

Affichage des attributs

L'utilisateur n'a pas toujours besoin de voir tous les attributs des balises. Il a donc la possibilité de masquer les attributs qui ne l'intéressent pas.

Affectation de style aux éléments

Dans un souci de convivialité de l'interface, il est intéressant de donner la possibilité à l'utilisateur de choisir le style (couleur, police, etc.) qu'il veut affecter aux différents types d'éléments.

4.1.2 Manipulation

Ajout d'un élément

Lorsque l'on veut ajouter un élément, on peut vouloir d'une part insérer un élément au sein de l'arbre existant et d'autre part ajouter un nouvel élément.

- Insertion au sein d'un arbre

Il s'agit d'ajouter les balises, ouvrante et fermante, associées à un élément que l'on veut insérer dans l'arbre. Ainsi, dans le Tableau 4-1, nous ajoutons la balise ouvrante de <name type=objet key=01> et la balise fermante </name> autour de la chaîne de caractères H2. Notons que cette chaîne pourrait être elle-même composée d'éléments comme dans le Tableau 4-2, où nous ajoutons l'élément <u>.

<u id=u33 who=S2> Tracer l'intersection de **<name type=objet key=O1>**H2**</name>** et <name type=objet key=02>H3</name>en rouge.

Tableau 4-1: Ajout de balises <name> dans un PcData

```
<u>
<seg id=c1u69seg2 type=mf2 n=70>Opération terminée</seg>
<seg id=c1u69seg3 type=sentence n=71>Veuillez désigner la
partie à supprimer</seg>
</u>
```

Tableau 4-2: Insertion d'une balise <u>

- ajout d'un fils à un élément

Dans ce cas, l'utilisateur veut insérer un fils à une position précise par rapport à un élément donné (de la première position à la dernière position). Dans le Tableau 4-3, un élément <seg> est inséré comme premier fils de l'élément <u>. Il veut également se voir proposer le type d'élément qu'il peut insérer.

```
<u>
<seg id=c1u69seg1 type=ma3 n=69>Opération en cours</seg>
<seg id=c1u69seg2 type=mf2 n=70>Opération terminée</seg>
<seg id=c1u69seg3 type=sentence n=71>Veuillez désigner la
partie à supprimer</seg>
</u>
```

Tableau 4-3: Insertion d'une balise <seg>

Suppression d'un élément

L'élément sélectionné est supprimé ainsi que tous les éléments imbriqués.

Suppression de balises

La suppression de balises correspond au retrait des balises, ouvrante et fermante, entourant un élément. Il s'agit de l'inverse de la fonctionnalité d'insertion d'un élément au sein d'un arbre.

Rechercher-remplacer les occurrences d'un type de tag

Tous les tags d'un nom donné sont remplacés par un autre tag.

Ajout d'un attribut

Un attribut et sa valeur sont ajoutés dans la balise. Une liste des attributs déjà placés sous une balise de même type dans le document est proposée à l'utilisateur.

Suppression d'un attribut

L'attribut et sa valeur sont supprimés de la balise concernée.

Modification d'un attribut

Le nom ou la valeur de l'attribut peuvent être changés.

Calcul des occurrences d'une balise

Cette fonctionnalité calcule le nombre d'occurrences de chaque balise du fichier SGML.

Copier-couper-coller des parties de dialogues

Les fonctions couper et coller traitent un élément tout entier qui peut être ensuite collé par la fonction de même nom comme fils à un autre élément.

Gestion des références

La gestion des références aux actions et aux objets doit permettre à l'utilisateur d'ajouter, de supprimer et de modifier aussi bien la balise source que la balise cible de la référence. Cette opération doit se réaliser en un minimum d'actions. Le lien entre la cible et la source doit être automatique.

4.2 Fonctionnalités pour l'éditeur SGML sous forme d'arbres.

La notion de noeud dans l'éditeur SGML sous forme d'arbres est la même que celle d'éléments utilisée dans le point concernant les fonctionnalités de l'éditeur de dialogues. Le terme noeud est spécifique aux arbres.

4.2.1 Filtrage

Changement de l'orientation de l'arbre

Les différents utilisateurs ont chacun leurs habitudes quant à la façon dont ils veulent voir leurs arbres affichés, soit horizontalement, soit verticalement.

Cacher - montrer des parties d'arbres

Afin de clarifier la présentation d'un arbre, il est intéressant de pouvoir cacher certaines parties de celui-ci et inversement de les faire réapparaître.

Cacher - montrer les attributs d'un nœud

Même si l'affichage de tous les attributs de chaque nœud permet une information complète, il rend malaisé la lecture et la manipulation de l'arbre. Il est donc important de pouvoir choisir les nœuds pour lesquels on désire ou non voir afficher les attributs.

Afficher le contenu du document sous forme textuelle

Les utilisateurs, surtout au début de l'utilisation de nouveaux logiciels, aiment pouvoir apprécier les modifications faites à l'aide de leur outil et vérifier ainsi si ce dernier a bien eu le comportement souhaité.

Filtrer les nœuds à afficher

Lors de travaux particuliers, les informations importantes peuvent se trouver noyées dans la somme de tout ce qui se trouve dans le fichier. L'utilisateur voudra donc réduire la masse d'informations qu'il veut voir affichée.

Styliser l'affichage de l'arbre

Si l'utilisateur veut pouvoir filtrer les informations à afficher. Il veut également être capable de faire ressortir certaines informations par l'utilisation de styles (couleurs, images, polices, etc.).

4.2.2 Manipulation

Effacer un nœud

Il s'agit de supprimer une balise du fichier ainsi que toutes celles incluses au sein de cette première.

Ajouter un nœud

Non seulement il est indispensable de pouvoir ajouter une balise, mais il faut également aider l'utilisateur en lui proposant les balises autorisées à cet endroit. Cependant, comme nous l'avons expliqué plus haut, cela nécessite un *Parser* de DTD, ce qui est difficilement réalisable. Afin d'aider l'utilisateur, il est toujours possible de proposer les balises qui sont déjà positionnées à un endroit semblable au travers du fichier manipulé.

Modifier la valeur d'un nœud[36]

La modification de la valeur d'une balise au milieu d'un arbre peut rapidement corrompre le fichier. On n'autorisera donc que la modification de nœuds feuilles de type "données " (PCDATA).

Ajouter - Modifier - Effacer un attribut

Dans ce cadre, il faut permettre l'ajout, la modification et la suppression d'un attribut. Lors de l'ajout, il est nécessaire aussi bien d'initialiser la valeur de l'attribut que de proposer l'ensemble des attributs déjà placés sous le type de balise courante.

Copier - couper - coller de parties d'arbres

La copie, le coupage et le collage de parties d'arbres doivent être possibles.

Gestion des références

Le suivi des liens internes (ex : balise TEI <ptr>) et externes (ex : balise TEI <xptr>) au fichier doit être rendu possible.

[36] On entend ici la valeur du *tag*, car la DILIB considère de la même manière un élément terminal de type "donnée " (PCDATA) et le *tag* d'un élément non terminal.

Vérification de la correspondance du document par rapport à une DTD spécifique

La vérification de la correspondance du document à l'aide d'un *Parser* est nécessaire puisque nous ne pouvons le faire au fur et à mesure. Nous devons donc permettre le choix d'un *Parser* ainsi que de ses paramètres.

4.2.3 Spécificités liées aux arbres adjoints

Il n'y a pas de spécificités propres aux arbres adjoints quand on les compare aux arbres SGML en général. En effet, la chose principale dont le manipulateur d'arbre adjoint voudra disposer est un style d'affichage particulier pour les différents nœuds de son arbre, ce style tenant compte de la valeur des attributs. Ces attributs devront influencer non seulement le style de l'affichage du nœud qui les contient, mais également le style d'autres nœuds qui pourront être leur fils, descendants ou ancêtres.

On notera cependant que, dans le cadre des recherches poursuivies au C.R.I.N., il serait intéressant de donner la possibilité de répéter une fonctionnalité donnée, à intervalle régulier. En effet, cette fonctionnalité se révèle très intéressante dans le cadre de tests de programmes effectuant l'analyse de phrases. Les résultats de ce programme devant être visualisés à chaque étape, il est fastidieux de devoir recharger un même fichier plusieurs fois. Aussi, il est plus facile de demander à l'application de recharger le document à intervalles réguliers.

4.3 Fonctionnalités liées à d'autres applications

Avant d'essayer de rassembler les fonctionnalités communes aux deux applications examinées, il est opportun de se demander si nous ne pouvons pas généraliser notre démarche. La question que nous devons nous poser est de savoir s'il existe d'autres applications de même type et qui nécessiteraient le même genre de fonctionnalités.

A titre exploratoire, citons l'exemple d'une application d'aide à la conception de lexiques ou de dictionnaires. En effet, nous pouvons apparenter le travail sur un dialogue à celui effectué sur un lexique. Les différences entre ces deux applications sont indépendantes du logiciel d'édition. Seuls les éléments manipulés diffèrent. La manière d'utiliser l'éditeur est la même. Ce qui diffère entre ces deux approches, c'est la DTD utilisée. De là à dire qu'il n'y a pas de différences entre une application SGML, dite

scientifique[37], et une application grand public, il n'y a qu'un pas que nous hésitons à franchir. En effet, il nous paraît assez audacieux de classer un éditeur d'arbres SGML dans la catégorie application grand public. Qui donc voudrait manipuler son document SGML de la sorte ?

Citons encore un exemple d'aide à la conception de documentation. Ici, la personne, qui désire construire une documentation technique sera confrontée aux mêmes types de problèmes. Elle désirera créer un lexique qui lui permette d'accéder rapidement aux différents éléments de l'objet décrit. Elle voudra le faire non seulement à l'aide d'un lexique général, mais aussi au sein même du texte où une référence vers d'autres éléments que celui décrit est envisageable. Cette personne souhaitera également structurer ses descriptions afin de disposer d'une présentation commune et de permettre un travail automatisé ou semi-automatisé. Une fois encore, seule la DTD qui décrit le document technique diffère des applications d'édition de dialogues ou de lexiques.

La question que nous sommes en droit de nous poser maintenant est la suivante : quelle est la réelle différence entre l'éditeur d'arbres et l'éditeur textuel de documents SGML? Pour pouvoir répondre à cette question, examinons quelles sont les fonctionnalités communes à ces applications.

4.4 Fonctionnalités communes

La Figure 8 met en évidence les fonctions qui sont communes aux deux éditeurs tant au niveau de l'interface que du filtrage, de l'exportation et de la manipulation. Autour de ce noyau commun, viennent se greffer des fonctionnalités propres à chaque éditeur.

Dans cette figure, nous présentons la même découpe en niveaux qu'au chapitre précédent. Nous y avons cependant ajouté un niveau interface qui prend en compte les fonctionnalités propres à l'interface graphique.

L'interface est évidemment un point important de différenciation. Les deux interfaces ont chacune leurs spécificités dont la plus importante est la représentation du fichier SGML sous forme de texte ou sous forme d'arbres. Cependant, il existe un certain nombre de fonctionnalités communes telles que la gestion des fichiers ou celle des menus.

[37] i.e. à l'usage exclusif de la recherche scientifique.

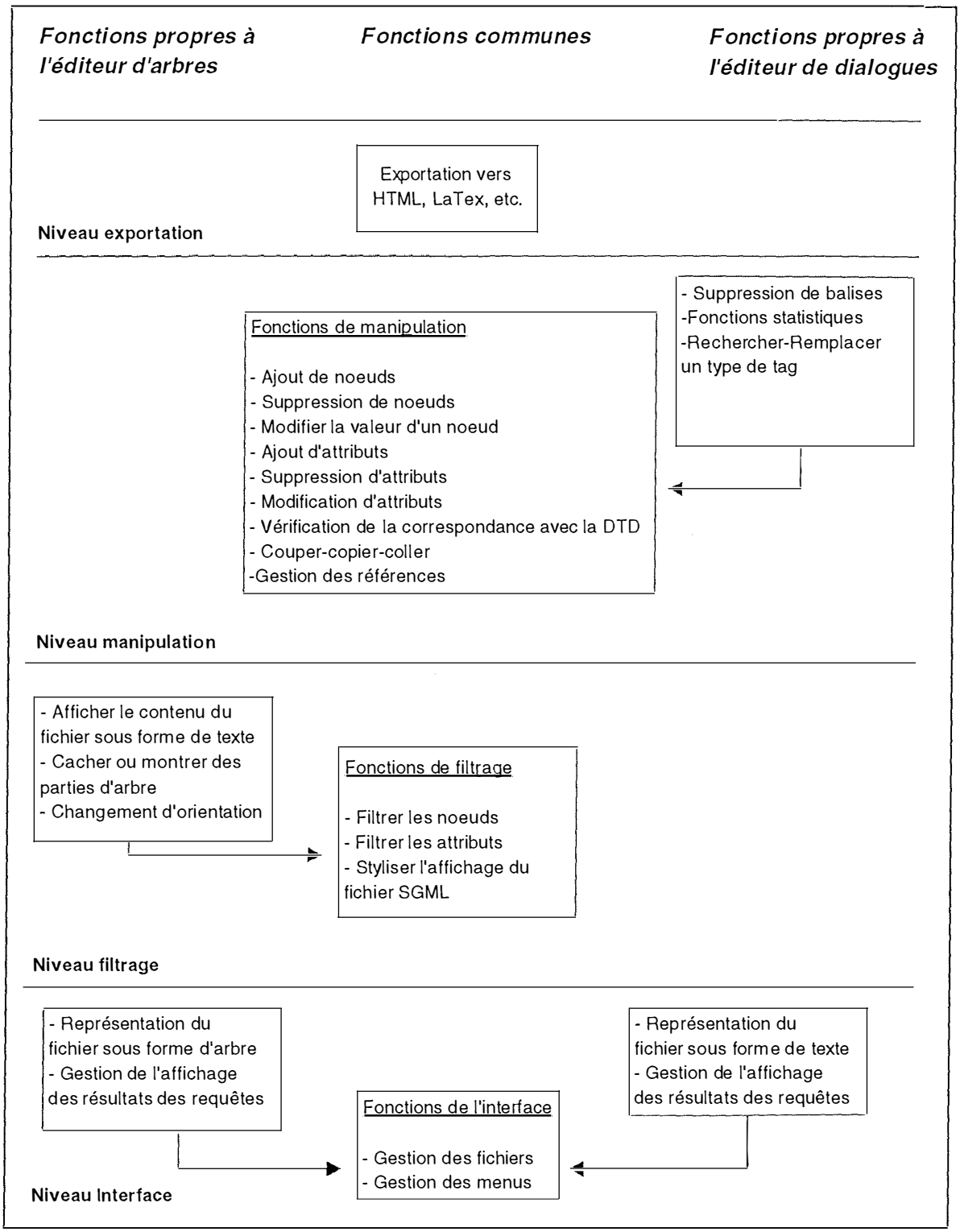

Figure 8 - Découpe en niveau des fonctionnalités

La partie interface gère également le retour visuel des requêtes de l'utilisateur suivant l'éditeur utilisé. Ainsi, l'ajout d'un noeud SGML sera représenté par une nouvelle

branche dans l'éditeur d'arbre, tandis que l'éditeur de dialogues ajoutera des balises ouvrantes et fermantes dans le texte.

Les fonctionnalités communes au niveau du filtrage et de la manipulation effectuent un travail identique sur le fichier SGML, mais le résultat visible à l'écran est différent selon qu'il s'agit de l'éditeur d'arbres ou de dialogues. Ainsi, les résultats de ces fonctions communes seront traités différemment pour s'intégrer dans l'éditeur utilisé.

4.5 Paradigme de ces fonctionnalités

On dégage, lors du développement de telles applications, un paradigme basé sur la trilogie «*données structurées – fonctions élémentaires sur les données - transparence lors de la visualisation* ». En effet, l'utilisateur qui travaille avec ces logiciels le fait pour des raisons précises. Le choix de SGML pour la structuration de ses données a été réfléchi et dépend le plus souvent d'un choix de l'organisation plutôt que d'un choix personnel de l'utilisateur.

D'autre part, l'application utilisée, d'un point de vue conceptuel, sera décomposée en deux parties. Une première partie vise l'accès aux données et leur manipulation. Elle est composée de fonctions élémentaires permettant d'ajouter un élément dans le fichier ou encore de copier une partie d'arbre. Ces données sont extrêmement structurées, la position des balises, leur nature, celle de leurs attributs sont définies très précisément. Sans vérification, une simple manipulation peut corrompre le fichier. Les règles peuvent être nombreuses et la connaissance de l'ensemble de celles-ci n'est accessible qu'aux seuls experts de la DTD utilisée.

Une utilisation sporadique ne permet pas à un utilisateur d'accéder à la connaissance de l'entièreté des règles. La cohérence du document doit donc être assurée par l'application elle-même. Ceci peut être fait par l'utilisation d'un *Parser* de DTD qui permettra, idéalement, la vérification de la cohérence après chaque action de l'utilisateur et, mieux encore, proposera à l'utilisateur les éléments dont il a besoin.

La deuxième partie de cette décomposition de l'application permet la visualisation du document manipulé. Elle doit permettre à l'utilisateur de manipuler un document sans se préoccuper des aspects techniques du langage. A l'extrême, un utilisateur n'ayant que peu de connaissances de SGML devrait pouvoir être capable de manipuler un document à

l'aide de l'éditeur, tout en ne connaissant que les concepts décrits dans la DTD et non leur forme. Cette interface doit être transparente pour l'utilisateur.

Les manipulations de données demandées par une telle interface résultent de la somme de nombreuses manipulations élémentaires. La suite de ces manipulations de base sont traduites par des fonctions. Ces dernières peuvent être amenées à modifier, de manière importante, la structure d'un document et à modifier le sens apporté à la donnée élémentaire manipulée. La cohérence du document doit être préservée.

Dès lors, le paradigme apparaît dans la dualité entre la nécessité de non corruption d'un document - et donc des données structurées qu'il contient - qui est implémentée à l'aide de fonctions simples et le besoin de manipulations complexes exigé par une interface transparente.

4.6 Conclusion

Les fonctionnalités que nous venons de décrire permettent de faire ressortir les besoins rencontrés par les utilisateurs d'applications générales d'édition de documents SGML. Certaines de ces fonctionnalités, demandent aux développeurs un travail important et la conception d'outils complexes. Il en va ainsi des fonctionnalités permettant de rendre l'interface « intelligente ».

La conception d'un outil intégrant les deux applications est possible, étant donné les nombreuses similitudes existant entre elles.

5. Les outils développés

Nous avons vu que, dans la plupart des cas, les fonctionnalités nécessaires aux applications particulières peuvent s'inscrire dans un cadre plus général. Nos outils sont des éditeurs SGML généraux, qui peuvent être utilisés dans le cadre d'applications particulières.

La conception d'une interface intelligente impose la réalisation d'un *Parser* de DTD, travail assez compliqué vu la taille et surtout la complexité que peuvent atteindre certaines DTD. La solution "à moindre coût" est de supputer que l'utilisateur sait ce qu'il fait et de n'effectuer aucune vérification de cohérence du document par rapport à sa DTD.

Cependant, il est dangereux de laisser l'utilisateur seul responsable de la cohérence de ses actions. En effet, personne n'est à l'abri d'une petite erreur. Il faut donc proposer à l'utilisateur la possibilité de vérifier, a posteriori, la validité de son document. Cette validation demande également l'utilisation d'un *Parser*. Cependant, elle a l'avantage de pouvoir utiliser un *Parser* qui a déjà été développé et qui ne doit pas être intégré à l'éditeur lui-même. En effet, il existe sur le marché un nombre non négligeable de *Parser*. Citons en exemple `sgmls` ou `nsgmls`, tous deux disponibles pour des plates-formes `Unix`.

Nous proposerons donc la possibilité d'exécuter, à partir de l'éditeur lui-même, le *Parser* de son choix avec les paramètres de son choix.

Les deux applications ont été développées en deux couches. La première, l'accès aux données, est réalisé en `C` et consiste en un ensemble d'exécutables qui sont appelés depuis la partie interface écrite en `TCL/Tk`[38]. Les paramètres transmis aux exécutables permettent de spécifier le fichier source, la position de l'élément, sa valeur, et éventuellement un fichier destination.

Dans le cadre de ces deux applications, nous différencions deux types de fonctions : les fonctions élémentaires et les fonctions complexes. Les fonctions élémentaires effectuent des actions simples sur les éléments ou parties d'éléments (*tags*, attributs, etc.) des documents SGML. Les fonctions complexes sont celles qui sont attendues par l'utilisateur. Elles nécessitent l'exécution de plusieurs fonctions élémentaires.

[38] i.e. `Tcl 7.5, tk4.1, tktree`

Les exécutables permettent d'effectuer des manipulations sur un fichier SGML. Ces derniers ont été réalisés en C et séparés l'un de l'autre plutôt que placés dans un fichier unique et ce pour deux raisons. La première est de limiter le nombre de paramètres associés à chacun d'entre eux. La seconde est de faciliter une éventuelle interface pour le Web. En effet, l'accès au Web est possible par la seule modification de l'interface. Notons qu'il existe un module `Tcl` qui peut être exécuté à partir d'un *browser* tout comme le permet `JavaScript`.

Les fonctions élémentaires qui sont utilisées pour ces manipulations sont tirées de la librairie DILIB.

La DILIB a été construite au C.R.I.N. par Jacques Ducloy et Patrice Bonhomme. Elle contient un ensemble de fonctions C donnant la possibilité de manipuler les éléments d'un document SGML. Elle permet, entre autres, d'ouvrir, de sauver un document SGML, d'ajouter et retirer des noeuds, des attributs. Elle permet aussi de couper, copier des parties d'arbres. Elle permet également de trouver le énième fils, le frère suivant ou précédent et le père du noeud courant, etc. Cependant, rien n'est prévu, ni pour référencer un noeud précis de l'arbre, ni pour les fonctionnalités complexes que nous avons décrites dans le chapitre précédent.

Le filtrage des données, commun aux deux applications, est effectué à l'aide d'une feuille de style. La feuille de style permet à l'utilisateur de formater l'affichage de son document, ainsi que d'y cacher des éléments. La description de cette feuille de style, commune aux deux applications, se trouve au point 5.3.

La seconde couche qui concerne l'interface est propre, en grande partie, aux applications. Nous pouvons cependant souligner que certains modules sont communs aux deux applications. Ils ne diffèrent que lors des appels de fonctions, les paramètres étant différents. Il en est ainsi pour les modules de gestion de menu, de gestion du système d'onglets, la gestion de l'aide, la gestion des fichiers.

Nous allons dans les deux points suivants (5.1 et 5.2) décrire les logiciels que nous avons eu l'occasion de développer.

5.1 Editeur de Textes

L'interface de l'éditeur se décompose en deux parties. La première partie contient le fichier SGML qui a été ouvert. L'affichage du fichier correspond aux caractéristiques de visualisation demandées par l'utilisateur : le choix des balises à masquer, les couleurs à

attribuer aux balises. Cette première partie permet d'effectuer des modifications sur le fichier.

La seconde partie donne la possibilité à l'utilisateur d'associer un fichier source au fichier SGML. Par fichier source, il faut comprendre un fichier de n'importe quel type susceptible de faciliter le travail de l'utilisateur sur le fichier SGML. Ainsi, par exemple, un utilisateur aimera consulter le document sous sa forme non codée. Cependant, si l'utilisateur n'a pas besoin de cette seconde partie, il pourra ne pas la faire apparaître.

Les différentes fonctionnalités offertes par l'éditeur sont reprises dans les menus suivants :

- File Menu
- Edit Menu
- Node Menu
- Attributes Menu
- PcData Menu
- Statistic Menu
- Help Menu

A) File Menu

Le Tableau 5-1 décrit les fonctions classiques de gestion des fichiers.

Nom action	fonction
Open	Ouvrir un fichier SGML sélectionné par l'utilisateur. Le document SGML apparaît selon les caractéristiques définies dans la feuille de style pour chaque élément (l'affichage a lieu dans la zone de texte supérieure).
Open Source	Ouvrir un fichier sélectionné par l'utilisateur qui normalement est le source du document SGML traité. La raison d'être de cette fonctionnalité est de permettre à l'utilisateur d'avoir sous les yeux à la fois le document source et le document SGML (l'affichage a lieu dans la zone de texte inférieur).
Save	Enregistrer les modifications effectuées dans le fichier SGML.
Save As	Enregistrer le fichier SGML sous un nouveau nom.
Close	Fermer le fichier courant et demander s'il faut enregistrer les modifications.
Revert	Remettre le fichier dans son état initial. Toutes les modifications sont annulées.
Split	Diviser l'écran en deux parties, l'une pour le document SGML, et l'autre pour le document source.
Unsplit (keep this)	La zone de texte dans laquelle le curseur de la souris se trouve est mise en plein écran.
Unsplit (keep other)	La zone de texte dans laquelle le curseur de la souris ne se trouve pas est mise en plein écran.
Quit	Quitter l'application et demander si les fichiers modifiés doivent être enregistrés.

Tableau 5-1 - Commandes de gestion des fichiers

B) Edit menu

Le menu d'édition offre les possibilités de `couper - copier - coller` et de recherche.(Tableau 5-2)

nom action	fonction
Cut	Couper un élément sélectionné.
Copy	Copier un élément sélectionné.
Paste	Ajouter un fils à un élément existant, le fils étant l'élément contenu dans le *clipboard* à la suite d'un *Cut* ou d'un *Copy*.
Undo	Annuler la dernière modification.
Search	Rechercher les occurrences d'une balise ou d'un attribut.
Replace All Tags	Remplacer toutes les balises d'un type donné par un autre type.

Tableau 5-2 - Commandes d'édition

C) Node Menu

Le Tableau 5-3 contient toutes les fonctions destinées à manipuler et à changer la structure du document SGML.

Nom action	fonction
Add Node	Encadrer un élément existant par une nouvelle balise.
Add Child	Ajouter un fils à un élément courant.
Delete Element	Supprimer un élément tout entier.
Add Tag	Ajouter une balise autour d'une partie de texte sélectionné.
Delete Tag	Effacer uniquement la balise ouvrante et fermante d'un élément.

Tableau 5-3 - Commandes de manipulation de la structure du fichier

D) Attributes Menu

Le Tableau 5-4 décrit les actions possibles sur les attributs d'une balise.

Nom action	fonction
Add Attribute	Ajouter un attribut dans une balise.
Delete Attribute	Effacer un attribut d'une balise.
Modify Attribute	Modifier la valeur d'un attribut.

Tableau 5-4 - Commandes de manipulation des attributs

E) PcData Menu

Le menu `PcData` offre à l'utilisateur une fonction permettant de modifier le contenu d'un `PcData`.

F) Statistic Menu

La fonctionnalité associée au menu statistique permet de calculer et d'afficher le nombre d'occurrences de chaque type de balise présente dans le fichier SGML.

G) Help Menu

L'utilisateur trouvera dans le menu Aide une présentation générale de l'éditeur ainsi qu'une explication de ses différentes fonctionnalités. Chaque fonction est détaillée et accompagnée d'un exemple.

5.2 Editeur d'arbres

L'interface graphique [PIRE, 97] (Figure 9) de l'application est divisée en deux parties, l'une permettant la visualisation, sous forme d'arbres, des fichiers ouverts, et l'autre permettant d'éditer les attributs et *tags* sélectionnés dans la première partie.

La partie visualisation permet de manipuler l'arbre représentant le fichier ouvert, à l'aide de la souris et de raccourcis clavier[39]. Un système d'onglets permet le passage d'un fichier ouvert à un autre. Les actions permises sont reprises dans les menus de l'application.

Les différentes menus sont :

- File Menu
- Edit Menu
- Tree Menu
- Node Menu
- Attributes Menu
- Option Menu
- Windows Menu
- Help Menu

[39] Combinaison de touche déclenchant une fonctionnalité.

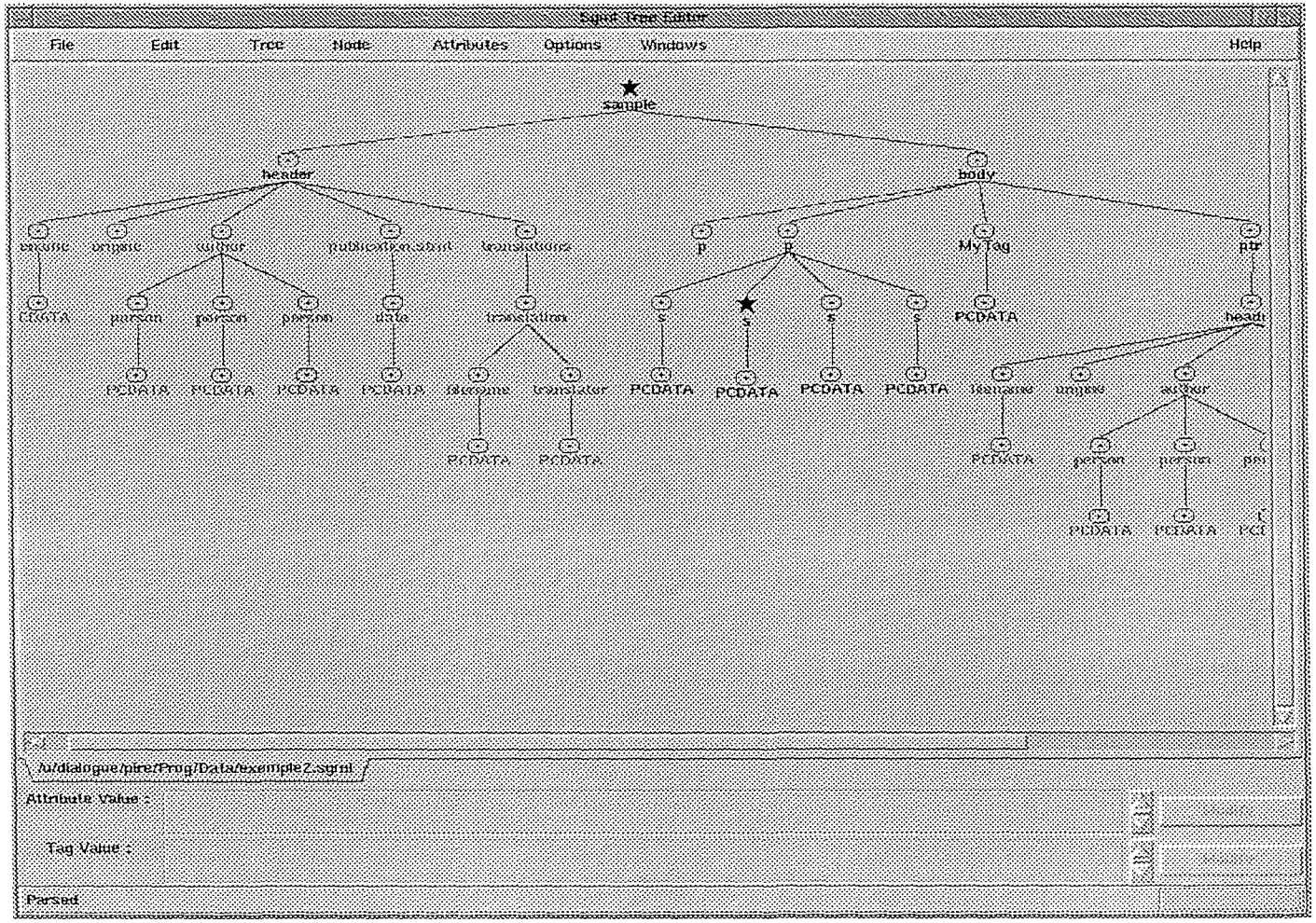

Figure 9 - Interface de l'éditeur d'arbres

A) File Menu

Le Menu fichier (File Menu) rassemble les fonctionnalités liées à l'ouverture, la fermeture, l'impression des documents. Les différentes fonctionnalités sont expliquées dans le Tableau 5-5.

Nom action	Fonction
New	Créer un nouveau fichier.
Open	Ouvrir un fichier SGML existant.
Save	Sauver le fichier SGML courant.
Save As	Sauver le fichier SGML courant sous un nouveau nom.
Close	Fermer le fichier SGML courant après avoir proposé de le sauver en cas de modification.
Revert	Remettre le fichier dans l'état dans lequel il était lors de l'ouverture ou de la dernière sauvegarde.
Print	Formater l'impression et imprimer l'arbre courant.
Quit	Quitter l'application, après avoir proposé la sauvegarde des fichiers modifiés.

Tableau 5-5 - Commandes du menu fichier

B) Edit Menu

Le menu d'édition permet la manipulation de morceaux d'arbres à l'aide des notions bien connues *Cut*, *Copy* et *Paste*, mais il permet aussi de « copier - coller » des références. On peut également y annuler l'opération précédente (Tableau 5-6).

Nom action	Fonction
Undo	Défaire la modification précédente.
Cut	Couper le sous-arbre dont la racine est le noeud sélectionné et le placer dans le *clipboard*.
Copy	Copier le sous-arbre dont la racine est le noeud sélectionné et le placer dans le *clipboard*.
Paste	Insérer, à la position choisie par rapport au noeud sélectionné, le contenu du *clipboard*. Les différentes positions sont : avant le noeud sélectionné (*before*), après (*after*), en tant que premier fils (*first child*) ou comme dernier (*last child*).
Copy reference	Copier la référence du noeud sélectionné.
Paste reference	Insérer à la position choisie (voir *Paste*) une balise `<xptr>` ou `<ptr>` si la référence est en dehors du fichier courant ou pas, respectivement.

Tableau 5-6 - Commandes du menu édition

C) Tree Menu

Le menu Arbre rassemble les fonctions manipulant l'arbre entier. Les commandes sont reprises dans le Tableau 5-7.

Nom action	Fonction
Clear Tree	Effacer l'arbre de l'écran.
Toggle layout	Changer la direction d'affichage de l'arbre (verticale ou horizontale).
Center	Centrer l'arbre au milieu de la fenêtre.
Redraw	Redessiner l'arbre.

Tableau 5-7 - Commandes du menu arbre

D) Node Menu

Le menu Noeud rassemble des commandes propres à la manipulation des noeuds (Tableau 5-8).

Nom action	Fonction
Add	Ajouter un noeud à une position choisie par rapport au noeud courant (*Last Child*, *First Child*, *Before*, *After).* La liste des noeuds déjà placés dans le fichier à un endroit similaire est proposée à l'utilisateur.
Delete	Effacer le sous-arbre dont la racine est sélectionnée.
Expand/Collapse subtree	Afficher les fils du noeud courant ou les cacher s'ils sont déjà visibles.
Expand all	Afficher tous les descendants du noeud courant.

Tableau 5-8 - Commandes du menu noeud

E) Attributes Menu

Le menu attribut rassemble les commandes de manipulations d'attributs (Tableau 5-9).

Nom action	Fonction
Show/Hide attributes	Afficher ou cacher la liste des attributs du noeud sélectionné.
Add	Ajouter un attribut au noeud courant.
Remove	Effacer l'attribut sélectionné.

Tableau 5-9 - Commandes du menu attributs

F) Option Menu

Le Menu option permet de modifier les paramètres propres à l'application tels que :

- Les répertoires par défaut;
- Les variables par défaut (nombre maximum de fichiers ouverts à la fois, *layout*, nombre de l'arbre de niveau à afficher, nom du *Parser*, ...);
- Les fonctions à déclencher à intervalles réguliers;
- Nom de la feuille de style à utiliser.

G) Windows Menu

Le menu Fenêtres permet d'ouvrir des fenêtres de visualisation ou de vérification de la cohérence du fichier.

Nom action	Fonction
Show SGML Code	Afficher la fenêtre contenant le code SGML correspondant à l'arbre affiché. L'utilisateur peut choisir entre une présentation mise en forme (indentée[39]) ou brute.
Parse SGML Code	Afficher le résultat de l'analyse de l'arbre par un *Parser*. Le *Parser* et le fichier d'en-tête sont définis dans le menu options.

Tableau 5-10 - Commandes du menu fenêtre

H) Help Menu

Le menu d'aide donne accès aux différents points d'entrée dans l'aide de l'application : Index, Commandes, Raccourcis clavier, Nouveautés, « A propos ». L'aide est construite à partir de pages HTML.

5.3 Feuille de style

Une feuille de style est un document SGML permettant la définition de styles. Un style permet d'afficher un document sous un format qui permet une lecture et une manipulation plus simples du document, par l'utilisation de couleurs, dessins, polices, épaisseurs de traits, etc. Une DTD est définie pour les feuilles de style. Elle permet de vérifier la validité d'une feuille de style, avant son application.

[39] En indentant, on place à une même distance de la marge droite du document les éléments de même niveau structurel.

Nous dénommerons dorénavant « élément courant », l'élément traité par l'application et auquel nous cherchons à appliquer un style.

La balise racine d'une feuille de style est <stylesheet>. Elle peut être composée des balises : <stylegrp>, <file>, <element> et <attribut>. selon la définition du Tableau 5-11.

```
<!ELEMENT STYLESHEET - - ( STYLEGRP?, ELEMENT*, FILE*, ATTRIBUTE*)>
 <!ATTLIST STYLESHEET name CDATA #REQUIRED >
```

Tableau 5-11 - Définition des élément principaux d'une fenêtre de style

La balise <stylesheet> possède un attribut name obligatoire, qui spécifie le nom de la feuille.

La balise <stylegrp> rassemble les styles prédéfinis, et ne peut contenir que des balises <style> (Tableau 5-12).

```
<!ELEMENT STYLEGRP - - (STYLE)*>
```

Tableau 5-12- Définition de la balise <stylegrp>

L'élément <style> (Tableau 5-13) ne peut contenir aucun autre élément et possède un attribut obligatoire qui définit le nom du style ainsi que des attributs caractérisant le style lui-même(Tableau 5-14).

```
<!ELEMENT STYLE - - empty >
<!ATTLIST STYLE %a.style;
              name CDATA #REQUIRED >
```

Tableau 5-13 - Définition de l'élément style

```
<!ENTITY % a.style ' bitmapcolor CDATA #IMPLIED
        checkedbitmap CDATA #IMPLIED
        color CDATA #IMPLIED
        display (yes I no) #IMPLIED
        font CDATA #IMPLIED
        line (solid I doted I dashed ) #IMPLIED
        linecolor CDATA #IMPLIED
        linewidth CDATA #IMPLIED
        style CDATA #IMPLIED
        textcolor CDATA #IMPLIED
        uncheckedbitmap CDATA #IMPLIED ' >
```

Tableau 5-14 - Définition des attributs de style

La dénomination des couleurs doit correspondre à celle acceptée par `Tcl` telles que: `red`, `blue`, `green`, encore `#c4c4c4`. Il en va de même pour les polices de caractères (*font*).

Les `bitmaps` doivent correspondre aux fichiers contenus dans le répertoire dédicacé[40].

L'attribut `display` permet de demander l'affichage ou non de l'élément.

Ayant ainsi défini les moyens de caractériser un format d'affichage, voyons comment associer un format à un contexte particulier du document SGML à l'aide de balises contextuelles. Ces balises font référence au contexte dans lequel se trouve chaque noeud du fichier SGML à afficher.

Lors de l'affichage d'un noeud courant, on vérifiera si l'application doit lui affecter un style particulier.

La balise `<element>` (Tableau 5-15) permet de fixer un format de style en fonction du `tag` de l'élément courant.

[40] Ce répertoire est un répertoire par défaut désigné dans le menu options de l'application.

```
<!ELEMENT ELEMENT    - - ( ELEMENT*, FILE*, ATTRIBUTE* )>
<!ATTLIST ELEMENT     %a.style;
                name     CDATA #IMPLIED
                style    CDATA #IMPLIED
                context  CDATA #IMPLIED >
```

Tableau 5-15 - Définition de la balise <element>

Imaginons que le noeud courant est une balise <p> et que dans le fichier de style la balise <stylesheet> ait comme fils une balise <element> dont l'attribut name est "p". Dès lors, les attributs de style associés à cette balise <element name = "p"> seront appliqués lors de l'affichage du noeud <p> courant. Si la balise <element> contient l'attribut style="style1", le style de nom "style1" sera appliqué à l'élément <p>, ainsi qu'à tous ses semblables.

Si un attribut de style (color, bitmaps, etc.) est défini d'une part dans le style <style name="style1"> et d'autre part dans la balise <element>, alors, c'est cette dernière définition qui prime.

L'attribut context permet de n'appliquer les attributs de style à l'élément courant que si ce dernier correspond à un certain contexte. Les termes acceptés pour définir le contexte sont : root, ancestor, child, descendant, next, previous et id. Ils reposent sur la définition des *ladders* de la TEI [GETEI, 97]. Ils sont interprétés de la manière suivante : si l'élément référencé est présent dans l'attribut context, alors les attributs de styles de cet élément sont appliqués au noeud courant.

Les balises insérées dans une autre balise de définition de style voient leurs attributs de style appliqués en priorité. Ainsi, l'exemple contenu dans le Tableau 5-16 sera interprété de cette façon : les noeuds <u> qui ont un fils <seg> seront affichés en bleu, et les autres noeuds <u> le seront en rouge.

```
<element name="u" color="red">
    <element name="u" context="child (1 seg)" color="blue">
    </element>
</element>
```

Tableau 5-16 - Exemple de priorité entre styles définis dans des balises imbriquées

La priorité entre deux balises de même niveau sera donnée à celle qui est placée en premier lieu sur la feuille de style. Ainsi, l'exemple introduit dans le Tableau 5-17 verra tous les éléments <u> affichés en rouge.

```
<element name="u" color="red">
</element>
<element name="u" color="blue">
</element>
```

Tableau 5-17 - Exemple de priorité entre styles définis dans des balises de même niveau

La balise `<file>` a les mêmes propriétés que la balise `<element>` mais vérifie le fichier auquel appartient le noeud courant (Tableau 5-18). En effet, un noeud affiché par l'application peut appartenir à un autre document grâce au mécanisme de liens hypertextuels.

```
<!ELEMENT FILE - - ( ELEMENT*, FILE*, ATTRIBUTE* )>

<!ATTLIST FILE %a.style;
        name CDATA #IMPLIED
        style CDATA #IMPLIED
        context CDATA #IMPLIED >
```

Tableau 5-18 - Définition de l'élément file

La balise `<attribute>` (Tableau 5-19) a les mêmes propriétés que la balise `<element>` mais est appliquée en fonction des attributs du noeud courant. Ainsi, l'exemple du Tableau 5-20 fera en sorte de ne pas afficher les noeuds qui ont un attribut `id`, quelle que soit la valeur de ce dernier et affichera la ligne reliant le noeud courant a son père en pointillés s'il contient un attribut type dont la valeur est égale à "1"[41].

[41] L'attribut `rel` permet de définir la relation entre l'attribut principal de la balise et sa valeur. Il peut prendre les valeurs : `eq| ne| lt | le | gt | ge`

```
<!ELEMENT ATTRIBUTE - - ( ELEMENT*, FILE*, ATTRIBUTE* )>

<!ATTLIST ATTRIBUTE %a.style;
          name CDATA #IMPLIED
          rel %v.rel #IMPLIED
          value CDATA #IMPLIED
          style CDATA #IMPLIED
          context CDATA #IMPLIED >
```

Tableau 5-19 - Définition de l'élément `attribute`

```
<attribute name="id" display="no">
</attribute>
<attribute name="type" rel="eq" value="1" line="doted">
</attribute>
```

Tableau 5-20 - Exemple d'utilisation de l'élément `attribute`

La définition complète de la DTD se trouve dans le Tableau 5-21.

```
<!DOCTYPE STYLESHEET [
<!ENTITY % a.style '
                bitmapcolor    CDATA  #IMPLIED
                checkedbitmap  CDATA  #IMPLIED
                color          CDATA  #IMPLIED
                display        (yes I no) #IMPLIED
                font           CDATA  #IMPLIED
                line           (solid I doted I dashed ) #IMPLIED
                linecolor      CDATA  #IMPLIED
                style          CDATA  #IMPLIED
                textcolor      CDATA  #IMPLIED
                uncheckedbitmap CDATA  #IMPLIED  '>
<!ENTITY % v.rel '(eqI neI lt I le I gt I ge)'>

<!ELEMENT STYLESHEET - - ( STYLEGRP?, ELEMENT*, FILE*,
ATTRIBUTE*)>
<!ATTLIST STYLESHEET    name          CDATA  #REQUIRED >

<!ELEMENT STYLEGRP   - - (STYLE)*>

<!ELEMENT STYLE      - - empty >
<!ATTLIST STYLE         %a.style;
                name          CDATA  #REQUIRED >

<!ELEMENT ELEMENT    - - ( ELEMENT*, FILE*, ATTRIBUTE* )>
```

```
<!ATTLIST ELEMENT       %a.style;
                name        CDATA #IMPLIED
                style       CDATA #IMPLIED
                context     CDATA #IMPLIED >

<!ELEMENT FILE      - - ( ELEMENT*, FILE*, ATTRIBUTE* )>
<!ATTLIST FILE          %a.style;
                name        CDATA #IMPLIED
                style       CDATA #IMPLIED
                context     CDATA #IMPLIED >

<!ELEMENT ATTRIBUTE  - - ( ELEMENT*, FILE*, ATTRIBUTE* )>
<!ATTLIST ATTRIBUTE   %a.style;
                name      CDATA #IMPLIED
                rel       %v.rel #IMPLIED
                style     CDATA #IMPLIED
                value     CDATA #IMPLIED
                context   CDATA #IMPLIED >
```

Tableau 5-21 - DTD complète définissant les feuilles de style

5.4 Exportation

L'idée sous-jacente à l'exportation est la mise en forme du contenu du document dans un format tel que HTML. Cela nécessite l'établissement de règles de correspondances entre les éléments et les structures des différents formats. Les recherches que nous avons faites pour y arriver nous ont conduits vers l'application de la norme DSSSL (*Document Style Semantics and Specification Language*).

DSSSL est un langage utilisé pour associer des règles de formatage avec les éléments d'un document structuré du type SGML ou ODA. Il est divisé en deux parties :d'une part, un langage de transformation d'arbres qui peut être utilisé pour réordonner des documents structurés avant de les présenter ; d'autre part, un processus de formatage qui associe des instructions de formatage à des noeuds spécifiques du document à présenter.

DSSSL se base sur un dérivé du LISP. Il nécessite la mise au point de documents permettant de décrire le type d'exportation ainsi que la réalisation d'un interpréteur permettant l'application de ces descriptions. Ce qui pourrait être le sujet d'un ou deux stages.

6. Conclusion

L'utilisation de plus en plus répandue de documents électroniques a donné naissance à des normes de codage. Dans ce travail, nous avons eu l'occasion de présenter les normes et langages utilisés au Centre de Recherches en Informatique de Nancy et plus précisément par l'équipe Dialogue. Le besoin de créer des outils permettant de visualiser et de manipuler des documents codés selon ces normes et langages s'est fait ressentir. C'est donc dans ce cadre que nous avons développé nos deux éditeurs.

L'éditeur de dialogues aussi bien que celui d'arbres ont pour but de faciliter la tâche des personnes qui doivent manipuler des documents électroniques codés selon la norme SGML. Ils doivent fournir à l'utilisateur une manière rapide et efficace de manipuler les documents. Ils automatisent des tâches que l'utilisateur devait auparavant effectuer « manuellement » dans un éditeur de textes classique. L'éditeur de dialogues travaille sous une forme textuelle tandis que l'éditeur d'arbres offre à l'utilisateur un outil de manipulation graphique.

Les deux applications sont structurées selon trois niveaux de fonctionnalités qui sont le filtrage, la manipulation et l'exportation. Les fonctionnalités de filtrage permettent à l'utilisateur de visualiser son document selon certaines caractéristiques de style qu'il peut adapter selon le type de travail à effectuer. Les fonctionnalités de manipulation apportent une aide et un support à l'utilisateur pour effectuer des opérations relativement complexes sur le fichier SGML.

Après avoir dégagé les différentes fonctionnalités communes aux deux éditeurs, nous nous sommes finalement demandés ce qui différenciait ces derniers. Une piste qui nous semble intéressante à suivre est celle de la distance existant entre l'interface et la représentation SGML sous-jacente. En effet, l'interface de l'éditeur d'arbres, de part sa forme graphique est beaucoup plus proche de l'application visée, à savoir l'édition d'arbres adjoints, que ne l'est l'interface de l'éditeur de dialogues. L'éditeur de dialogues est en réalité un éditeur général de documents SGML.

Le niveau exportation n'a pas été réalisé dans les applications étant donné sa complexité. Cependant, il devrait permettre à l'utilisateur de transformer le format de son document de manière automatique. Il nous semble que le langage DSSSL est une piste intéressante pour l'élaboration de ce niveau exportation.

7. Bibliographie

[BONHOM, 96] Bohomme Patrice, Bruneseaux Florence, Romary Laurent, Codage, Documentation et Diffusion de Ressources Textuelles, http :// www.loria.fr/~bonhomme/GutTei, juillet 96.

[BRUNE, 96a] Bruneseaux Florence, Dialogue Homme-Machine et TEI: codage de dialogues (GOCAD), CRIN, 1996.

[BRUNE, 96b] Bruneseaux Florence, Dialogue Homme-Machine et TEI: différents niveaux de codage pour un même dialogue, CRIN, février 1996.

[BRUNE, 96c] Bruneseaux Florence, Codage des références et coréférences dans les dialogues Homme-Machine, CRIN, décembre 1996.

[CES, 96] Ide Nancy, Corpus Encoding Standard : version définitive de la Norme CES développée dans les projets Eagles et MulText et MulText-East, http://www.cs.vassar.edu/CES/, 1996.

[CRIN, 97] home page du C.R.I.N., http://www.loria.fr/CRIN, juillet 1996.

[DIAL, 97] home page de l'équipe Dialogue, http ://www.loria.fr/CRIN/equipe/dialogue, juillet 1997.

[DUCROT, 72] Ducrot Oswald et Todorov Tzvetan, *Dictionnaire encyclopédique des sciences du langage*, Editions du Seuil, Paris, 1972.

[GETEI, 97] TEI Guidelines for Electronic Text Encoding and Interchange ,A Gentle Introduction to SGML, http://etext.virginia.edu/TEI.html, juillet 1997.

[GOOSS,] Goossens Michel and Saarela Janne, A Practical Introduction to SGML, CERN, http ://faust.irb.hr/~cern/www/publications/sgmlen/sgmlen.html, octobre 1996.

[GUT, 96] Cahier GUTenberg, numéro 24, numéro sur la TEI, ISSN 1140-9304, juin 1996.

[HYTIM, 97] Présentation du langage hytime, Http ://dmsl.cs.uml.edu/standards/hytime.html, août 1997.

[JAUME, 97] Jaume Pascal, Présentation de l'éditeur de dialogues, http://loria.fr/~jaume/editor/intro.html, janvier 1997.

[JOHAN, 95] Johansson Stig, The encoding of spoken texts, *Computers and Humanities*, 1995.

[JOSHI, 75] Joshi A., Levi L., Takahashi M., Tree Adjunct Grammars, *Journal of the Computer and System Sciences*, 1975.

[LOPEZ, 96] Lopez Patrice, La syntaxe dans les interfaces multimodales intelligentes à composante orale : Etude Bibliographique, Rapport de Stage ESIAL 3ème année d'informatique, 9 juillet 1996.

[PIRE, 97] Pire Arnaud, Présentation de l'éditeur d'arbre SGML ste, http://www.loria.fr/~pire/ste, janvier 1997.

[ODA, 97] Présentation du langage ODA http://www2.echo.lu/impact/oii/oiiold/docstand.html#ODA, août 1997.

Annexes

Annexe A Les mécanismes simples d'analyse

Le chapitre 15 de la TEI [GETEI, 97] décrit un ensemble d'éléments qui permet d'associer des éléments d'analyse et d'interprétation simples avec des éléments du texte. Le terme analyse représente toute interprétation sémantique ou syntaxique que l'encodeur veut associer à une partie de texte.

1 Les catégories de segment linguistique

Il s'agit d'introduire différents éléments permettant de représenter la segmentation d'un texte en catégories linguistiques. Ces dernières sont : *sentence* <s>, *clause* <cl>, *phrase* <ph>, *word* <w>, *morpheme* <m>, *characters* <c>. Ces éléments décrivant les catégories linguistiques sont des cas spéciaux de la classe générique <seg>. Ils disposent tous d'un attribut `type` (qui caractérise le type du segment) et d'un attribut `function` (qui caractérise la fonction du segment).

Dans le Tableau 1, la segmentation se décompose de la manière suivante :

- Le texte est découpé en *sentences*. L'élément <s> est utilisé pour segmenter un texte en une série de segments qui ne se chevauchent pas.
- Les *sentences* peuvent être découpées en *clauses*.
- les *clauses* peuvent être elles-mêmes découpées en *phrases*.

Cette découpe n'est pas la seule possible ; ainsi, un texte peut être directement segmenté en *clauses* ou en *phrases* sans inclure une segmentation de plus haut niveau.

```
<p>
 <cl type='finite declarative' function='independent'>
  <phr type=NP function='subject'>Nineteen fifty-four,
   <cl type='finite relative declarative' function='appositive'>when
    <phr type=NP function='subject'>I</phr>
    <phr type=VP function='predicate'>was eighteen years old</phr>
   </cl>
  </phr>
...
 <\cl>
<\p>
```

Tableau 1 : Segmentation d'un texte en catégories linguistiques

L'élément <s> peut être vu comme une abréviation du *tag* <seg type='s-unit'> à la différence que contrairement aux éléments <seg>, les éléments <s> ne peuvent être emboîtés les uns dans les autres.

L'attribut *type* de l'élément <s> correspond à l'attribut subtype de l'élément <seg>, ainsi le *tag* <s type='xxx'> doit être vu comme un synonyme du *tag* <seg type='s-unit' subtype='xxx'>.

Il en est de même pour les éléments <cl> et <phr> qui peuvent être vus respectivement comme <seg type=clause> et <seg type=phrase>.

La même interprétation peut être attachée aux éléments <w>, <m> et <c>. Les éléments <w> , <m> et <c> ont la même signification que l'élément <seg> avec un attribut type de valeur `w`, `m` ou `c`. Cependant quelques restrictions portent sur ces trois éléments.

- L'élément <w> peut seulement contenir les éléments <w> ,<m>, <c> et PcData.
- L'élément <m> peut seulement contenir les éléments <c> et PcData.
- L'élément <c> peut seulement contenir un PcData

Les restrictions portant sur ces trois éléments demandent une attention particulière lors de leur utilisation. Le Tableau 2 décrit la segmentation d'une occurrence d'un élément <mentioned> en un élément <w>. La première proposition est correcte étant donné que <w> contient un élément autorisé, à savoir un PcData. La seconde proposition, quant à elle, est incorrecte car l'élément <mentioned> ne peut être contenu dans <w>.

<mentioned>grandiloquent</mentioned>
⇓
proposition 1 : <mentioned><w>grandiloquent</w></mentioned>
proposition 2 : <w><mentioned>grandiloquent</mentioned></w>

Tableau 2 : Utilisation de l'élément <w>

Les éléments <w> et <m> peuvent avoir des attributs additionnels qui peuvent être utilisés dans des index ou dans diverses applications analytiques.

L'attribut lemma peut être ajouté à un élément <w> pour indiquer par exemple l'infinitif d'un verbe conjugué ou encore pour indiquer le nominatif d'un nom dans le cas de la langue latine.

De manière similaire, l'attribut baseform peut être ajouté à l'élément <m> pour indiquer la forme de base d'un morphème transformé.

2 Les éléments et <interp>

Les éléments et <interp> ainsi que leurs éléments de regroupement, <spanGrp> et <interpGrp>, permettent d'attacher des notes, des remarques analytiques dans les textes. Le rôle et les attributs de ces éléments sont décrit dans le Tableau 3.

	Associe une annotation interprétative à un morceau de texte. Ses attributs sont : value : identifie le phénomène annoté from : début du passage to : fin du passage Cet élément permet de ne pas ajouter dans le texte de nouvelles balises contrairement à <interp>.
<SpanGrp>	Rassemble des éléments
<interp>	permet une annotation interprétative qui peut être liée à un morceau de texte. Son seul attribut est value qui identifie le phénomène annoté. Il est possible de définir à l'avance les éléments qui permettront la description des différentes interprétations pouvant être associées aux éléments du document.
<interpGrp>	Rassemble des éléments interp

Tableau 3 : Description des éléments et <interp>

Ces quatre éléments partagent deux attributs communs :

- resp : indique le responsable de l'interprétation
- type : indique le type du phénomène noté dans le passage (image, character, theme, allusion, discourse type)

Le Tableau 4 donne un exemple de l'utilisation de l'élément .

```
<p id=MQp1s2p114>
<s id=MQp1s2p114s1> There was certainly a definite point
at which the thing began.</s>
<s id=MQp1s2p114s2>It was ... away.</s>
<s id=MQp1s2p114s3>There was a slow... atoms.</s>
<s id=MQp1s2p114s4>She felt the river ... sun.</s>
<s id=MQp1s2p114s5>Not for one second longer river afterwards.</s>
<span resp=DTL
      value='the moment'
       from= <s id=MQp1s2p114s2>
       to= <s id=MQp1s2p114s4>
<s id=MQp1s2p114s6>For during ... of humanity.</s>
</p>
```

*Tableau 4 : Utilisation de l'élément *

L'élément peut être placé directement en-dessous du morceau de texte auquel il est associé ou à la fin du texte dans un élément <spanGrp>.

L'élément peut également représenter une division structurelle assignée à un récit comme montré dans le Tableau 5.

```
<p id=P1>
<s id=S1>Sigmund ... country.</s>
<s id=S2>Sinfiotli ... sons.</s>
<s id=S3>Borghild, Sigmund's ... brother.</s>
<s id=S4a> But Sinfiotli... same woman..</s>

<span resp=TMA type='structural unit' value='introduction'
      from =S1 to=S3>
<span resp=TMA type='structural unit' value='conflict'
      from S4a>
```

*Tableau 5 : Divisions structurelles à l'aide de l'élément *

Si des éléments ont le même attribut resp ou *type*, ils peuvent être groupés grâce à l'élément <SpanGrp> comme dans le Tableau 6.

```
<spanGrp resp=TMA type='structural unit'>
   <span value='introduction'    from=S1    to=S3>
   <span value='introduction'    from=S4a>
</spanGrp>
```

Tableau 6 : utilisation de la balise <spanGrp>

La même analyse peut être faite avec les éléments <interp> et <interpGrp>. L'élément <interp> fournit des attributs pour enregistrer une catégorie interprétative, sa valeur ainsi que l'identité de l'interpréteur mais l'on n'indique pas quel passage du texte est interprété. De cette manière, les mêmes structures peuvent être associées à plusieurs passages du texte. L'association entre les passages du texte et les éléments <interp> se fait par l'intermédiaire de l'attribut *ana* comme dans le Tableau 7.

```
<p id=....>
<seg id=MQp1s2p114s3-5 ana=moment>
<s id=MQp1s2p114s3>There was a ...</s>
<s id= ...          >She felt ...</s>
</seg>
</p>
<interp id=moment resp=DTL value='the moment'>
```

Tableau 7 : Utilisation de l'élément <interp>

L'élément <interpGrp> permet de regrouper les éléments <interp> et est rattaché au texte au moyen d'attributs ana (Tableau 8) ou d'éléments <link> .

```
<p id=P1>
<seg id S1-S3 ana=intro>
<s id=S1>Sigmund ... country.</s>
<s id=S2>Sinfiotli ... sons.</s>
<s id=S3>Borghild, Sigmund's ... brother.</s>
</seg>
<s id=S4a ana=conflict> But Sinfiotli... same woman..</s>

<interpGrp resp=TMA type='structural unit'>
<interp id=intro value='introduction'>
<interp id=conflict value='conflict'>
</interpGrp>
```

Tableau 8 : Utilisation de l'élément <interpGrp>

Les éléments <link> peuvent être regroupés dans un élément <linkGroup>. Cette méthode décrite dans le Tableau 9 ne demande plus la présence de l'attribut *ana* dans la partie texte.

```
<linkGroup resp=TMA argTypes='text interpretation' extendArgs=N>
<link targets='intro      S1-S3'>
<link targets='conflict   S4a'>
</linkGrp>
```

Tableau 9 :Utilisation de l'élément <linkGrp>

2.1 Annotations linguistiques

Le terme annotation linguistique signifie toute annotation déterminée par l'analyse des caractéristiques linguistiques du texte, excluant la structure formelle du texte (la découpe en chapitres ou en paragraphes) et les éléments de description du texte (les circonstances de sa production). Il existe plusieurs méthodes pour marquer

ces annotations; nous allons en aborder deux. Ainsi dans le Tableau 10 on définit des codes correspondant à chaque classe de mots (par exemple 'At' pour article ou encore NN1 pour nom singulier) à l'aide des éléments <interp>. Dans le texte, les éléments <ptr> sont définis et pointent vers les valeurs contenues dans les éléments <interp>.

```
<s>The<ptr target=AT> victim<ptr target=NN1>'s<ptr target=GEN>
friends<ptr target=NN2> told<ptr target=VVD> police ...</s>
<interpGrp type='word classes'>
        <interp id=AT value=AT>
        <interp id=NN1 value=NN1>
                .....
</interpGrp>
```

Tableau 10 : Utilisation de l'élément <interp> pour les annotations linguistiques

Si le texte est complètement segmenté (Tableau 11), il n'y a aucune difficulté ; l'attribut *ana* associe une interprétation pour chaque segment.

```
<s type=sentence>
<w ana=AT >The</w>
<w ana=NN1>victim</w>
<m ana=GEN>'s</m>
<w ana=NN2>friends</w>
...
<\s>
```

Tableau 11 : Annotation linguistique pour un texte complètement segmenté

Annexe B Feature Structures[1]

1 Introduction

Une « *feature structure* » est une structure générale de données qui identifie et groupe les « *individual structures* ». Suite à leur généralité, les *features structures* peuvent représenter beaucoup de types d'informations différentes et de relations entre des morceaux d'information. Leur instanciation en SGML fournit un métalangage pour représenter l'analyse et l'interprétation de textes.

2 Elementary feature structures

2.1 Features with binary values

Les éléments fondamentaux d'un système de *feature structures* sont les éléments <f> et <fs>. L'élément <fs> analyse une collection de *features* et leurs alternances comme unité structurelle. Ses attributs sont définis dans le Tableau 12.

Attribut	Fonction
type	indique le type de la feature structure
feats	pointeur vers les features
rel	indique la relation entre la valeur indiquée dans l'élément et le contenu réel. Les valeurs possibles sont : • Eq: les contenus sont identiques • Ne: le contenu réel n'est pas le contenu donné • Sb: le contenu réel est dans le contenu donné • ns: le contenu réel n'est pas dans le contenu donné

Tableau 12 : Attributs de l'élément <fs>

L'élément <f> associe un nom à une ou plusieurs valeurs de type différent. Ses attributs sont définis dans le Tableau 13.

[1] [GETEI, 97], chapitre 16.

Attribut	Fonction
Name	Nom de la feature
org	Indique l'organisation des valeurs données : single, set, bag, list
fval	Pointe vers les attributs identifiant des valeurs de feature
rel	Indique la relation entre le contenu de la *feature* ou de celle pointée par l'attribut fval et le contenu réel. Les valeurs possibles sont les mêmes que celles de la balise <fs>

Tableau 13 : Attributs de l'élément <f>

Dans le Tableau 14, un élément <fs> contenant des éléments <f> avec des valeurs binaires permet de coder les spécifications de *feature-value* pour des segments phonétiques. L'élément <plus> fournit la valeur binaire plus pour une *feature* tandis que <minus> fournit la valeur binaire moins pour une *feature*.

```
<fs type='phonological segment'>
    <f name=consonantal><plus></f>
    <f name=vocalic>  <minus></f>
    <f name=voiced>   <minus></f>
    <f name=anterior>  <plus></f>
    <f name=coronal>   <plus></f>
    <f name=continuant> <plus></f>
    <f name=strident>  <plus></f>
</fs>
```

Tableau 14 : Codage de feature-value pour des segments phonétiques

3 Feature, Feature-Structure et Feature-Value Libraries

La manière d'encoder les *feature structures* à la manière du Tableau 14 peut s'avérer fastidieuse et donner naissance à d'énormes fichiers. Une solution moins lourde serait d'utiliser l'attribut feats de l'élément <fs> pour pointer vers une ou plusieurs *feature* de cet élément. Cette méthode pour encoder les *feature structures* suppose que les <f> sont assignés à un identifiant SGML unique et sont rassemblés dans des *feature libraries*. (<fLib>)

Les *feature structures* peuvent être rassemblées dans des *feature structures libraries* (<fsLib>).

L'attribut `fval` peut être utilisé pour pointer vers ses valeurs. Cette méthode pour encoder les *feature values* suppose que les éléments valeurs possèdent un attribut identifiant et sont rassemblés dans des *features-value libraries* (`<fvLib>`). Le Tableau 15 décrit les éléments `<fLib>`, `<fsLib>` et `<fvLib>`.

Elément	Fonction
`<fLib>`	Librairie des éléments feature `type`: type de la *feature library*
`<fsLib>`	Librairie des éléments de *feature structures* `type`: type de la *feature structures library*
`<fvLib>`	Librairie des éléments de *feature-value* `type`: type de la *feature-value library*

Tableau 15 : Feature, feature-structure et feature-value

Le Tableau 16 représente une *feature library* pour les caractéristiques phonologiques.

```
<fLib type='phonological features'>
   <f id=cns1 name=consonantal><plus> </f>
   <f id=cns0 name=consonantal><minus></f>
   <f id=voc1 name=vocalic>   <plus> </f>
   <f id=voc0 name=vacalic>   <minus></f>
   <f id=voi1 name=voiced>    <plus> </f>
   <f id=voi0 name=voiced>    <minus></f>
   <f id=ant1 name anterior>  <plus> </f>
   <f id=ant1 name=anterior>  <minus></f>
   <f id=cor1 name=coronal>   <plus> </f>
   <f id=cor0 name=coronal>   <minus></f>
   <f id=cntl name=continuant  <plus> </f>
   <f id=cnt0 name=contìnuant> <minus></f>
   <f ìd=strl name=strident>  <plus> </f>
   <f id=str0 name=strident>  <minus></f>
</fLib>
```

Tableau 16 : Feature library

Les *feature structures* qui représentent l'analyse de segments phonologiques peuvent être représentées de la manière décrite dans le Tableau 17. Cependant, ce tableau ne représente que quatre possibilités de combinaison de caractéristiques phonologiques or il existe, dans ce cas-ci, bien plus (à partir des 7 caractéristiques binaires définies, il existe 128 combinaisons possibles). Toutes les combinaisons ne sont pas valables, celles qui ont du sens peuvent être reprises dans un élément `<fsLib>` comme dans le Tableau 18. Un identifiant unique est associé à chaque combinaison.

```
<fs feats='cns1 voc0 voi0 ant1 cor1 cnt0 str0'></fs>
<fs feats='cns1 voc0 voi1 ant1 cor1 cnt0 str0'></fs>
<fs feats='cns1 voc0 voi0 ant1 cor1 cnt1 str1'></fs>
<fs feats='cns1 voc0 voi1 ant1 cor1 cnt1 str1'></fs>
```

Tableau 17 : Codage des feature structures

```
<fsLib id=fsl1 type='phonological segment definitions'>
   <fs id=t.df feats=`cns1 voc0 voi0 ant1 cor1 cnt0 str0'></fs>
   <fs id=d.df feats='cns1 voc0 voi1 ant1 cor1 cnt0 str0'></fs>
   <fs id=s.df feats='cns1 voc0 voi0 ant1 cor1 cnt1 str1'></fs>
   <fs id=z.df feats='cns1 voc0 voi1 ant1 cor1 cnt1 str1'></fs>
</fsLib>
```

Tableau 18 : Utilisation de l'élément <fsLib>

Les éléments du texte peuvent être liés aux *feature structures* de la manière décrite dans le Tableau 19. Un élément <linkGrp> associe les caractères dans le texte à leur représentation phonologique. En effet, l'attribut target de chaque élément <link> pointe vers une combinaison de caractéristiques de l'élément <fsLib>. L'élément <fsLib> détermine les combinaisons de caractéristiques acceptables. L'élément <fvLib> détermine les valeurs possibles pour les caractéristiques phonologiques. L'élément <fLib>, quant à lui, associe plusieurs identifiants à chaque *feature* selon le nombre de valeurs qu'elle peut prendre, ces valeurs étant définies dans l'élément <fvLib>. Dans le cas présent, chaque segment phonologique a deux identifiants, selon que lui sera attribuée la valeur plus ou minus.

```
        <text id=text1>
<body>
<s id=s1>
  <w id=slwl>
    <c id=s1w1c1>C</c>ae<c id=s1w1c2>s</c>ar</w>
  <w id=s1w2>
    <c id=s1w2c1>s</c>ei<c id=s1w2c2>z</c>e<c id=s1w2c3>d</c></w>
 <w id=s1w3>
    <c id=s1w3c1>t</c>rol</w>.
</s>
</body>
</text>

<fsLib id=fsl1 type='phonological segment definitions'>
   <fs id=t.df feats='cns1 voc0 voi0 ant1 cor1 cnt0 str0'></fs>
   <fs id=d.df feats='cns1 voc0 voi1 ant1 cor1 cnt0 str0'></fs>
   <fs id=s.df feats='cns1 voc0 voi0 ant1 cor1 cnt1 str1'></fs>
   <fs id=z.df feats='cns1 voc0 voi1 ant1 cor1 cnt1 str1'></fs>
</fsLib>

<linkGrp  type='phonological identification of characters'
              domains='fs11 txt1' targFunc='phonological.segment character'
       extendArgs=repeat>
           <link id=lt targets='s.df s1w3c1'>
    <link id=ld targets='z.df s1w2c3'>
    <link id=ls targets='s.df s1w1c1 s1w2c1'>
          <link id=lz targets='z.df s1w1c2 s1w2c2'>
</linkGrp>

<fvLib type='binary values'>
       <plus  id=b1>
       <minus id=b0>
</fvLib>

<fLib type 'phonological features'>
        <f  id=cns1 name=consonantal fVal=b1></f>
        <f  id=cns0 name=consonantal fVal=b0></f>
        <f  id=voc1 name=vocalic       fVal=b1></f>
        <f  id=voc0 name=vocalic       fVal=b0></f>
        ....
</fLib>
```

Tableau 19 : Exemple complet de description de caractéristiques phonologiques

Annexe C Codage des références

Cette réflexion sur les références dans les dialogues homme-machine a été effectuée à partir de l'étude du corpus GOCAD [BRUNE, 96c].

1 Qu'est-ce qu'une référence?

Reboul et Moeshler parlent de « référence » pour *désigner « la relation qui unit une expression de la langue (« expression référentielle ») en emploi dans un énoncé et l'objet dans le monde que cette expression désigne.* ». Si par exemple au cours d'un repas, un convive (A) s'adresse à une personne précise (B) : « . . passe moi le pain: », B devra être capable d'identifier l'objet désigné pour pouvoir le transmettre à A. Le syntagme nominal « le pain » fait donc référence à un objet précis du monde. B, ayant repéré l'objet désiré par A, lui passe en disant « le voilà ». Nous dirons alors qu'il y a coréférence entre le N « le pain » et le pronom « le » puisqu'ils désignent tous les deux le même objet du monde. Le codage devra donc, non seulement mettre en évidence les expressions référentielles mais il devra aussi permettre de noter les coréférences.

Florence Bruneseaux et Laurent Romary ont proposé une manière de coder les références conformément à la TEI (Tableau 20). Il s'agit de noter la référence par la balise <rs>. Les coréférences ultérieures étant analysées comme des renvois à un <rs> déjà codé et noté par la balise <ref>, le lien entre <ref> et <rs> étant réalisé par un attribut target.

```
<u id="u145" who="C"> Les surfaces S1 et S2  ne sont pas
        visualisées à l'écran.</u>
 <u id="u146" who="S7"> Les visualiser.</u>

                              ⇓

 <u id="u145" who="C"> <rs key="O1"> Les surfaces S1 et S2 </rs>
        ne sont pas visualisées à l'écran.</u>
 <u id="u146" who="S7"> <ref target="O1">Les</ref>visualiser.</u>
```

Tableau 20 : Proposition de codage des références avec la balise <rs>

La balise <ref> qui permet un renvoi à une expression source n'est pas suffisante pour appréhender tous les types de relations pouvant exister entre des objets. Voici quelques exemples où la balise <ref> s'avère insuffisante.

- Nous entrâmes dans un village. L'église... et tous les autres exemples d'anaphores associatives ;

- La plume du stylo, le tronc de l'arbre... et les relations « partie-tout » ;
- Le chien de Sophie (possession), les accords de Paris (locatif)... et l'ensemble des relations entre les N des groupes nominaux complexes et des [N de N].

La proposition de codage consiste à utiliser la balise <rs> pour repérer les éléments qui réfèrent, avec un attribut indiquant le type de référence (à un objet, une action, ...). La balise <link> précisera le type de lien qui existe entre les expressions référentielles c'est-à-dire coréférence, association, partie-tout ...

2 Types de référence

Le Tableau 21 propose six types de référence avec pour chacun la possibilité d'ajouter des sous-types. L'ensemble de ces types est assez général mais les sous-types peuvent varier en fonction du corpus et de l'utilisation qui en sera faite.

```
<rs type=objet subtype="animé/inanimé/cognitif"
        action subtype="?"
        personne subtype="?"
        propriété subtype="?"
        événement subtype="?"
        lieu subtype="?"
    key="X">
```

Tableau 21 : Types de référence

3 Référence à un objet

Dans beaucoup de cas, il est facile de repérer au niveau du texte les SN[2] désignant des objets. Il est difficile de dire si des syntagmes identiques réfèrent ou non à un même objet. C'est pourquoi, utiliser la balise <rs> pour marquer la première occurrence d'un objet et ensuite faire référence à cet objet pour des balises <ref> n'est pas satisfaisant. La solution décrite dans le Tableau 22 a donc été proposée. Les SN référentiels sont mis en évidence, mais ce codage est insuffisant car il ne permet pas de montrer la relation qui existe entre le SN « les surfaces S1 et S2 » et le pronom « les ».

[2] Syntagme nominal.

<u id="u145" who="C"> **Les surfaces S1 et S2** ne sont pas visualisées à l'écran.</u>
<u id="u146" who="S7"> **Les** visualiser.</u>

⇓

<u id="u145" who="C"> <rs type="objet" key= "O1">Les surfaces S1 et S2 </rs> ne sont pas visualisées à l'écran.</u>
<u id="u146" who "S7">
<rs type="objet" key="O2"> Les</rs>visualiser.</u>

Tableau 22: Codage pour la référence à un objet

4 Référence à une action

Un sujet peut faire référence à une action unique ou à un ensemble d'actions dans un but de reproduction ou d'annulation.

4.1 Référence à une action unique ou à son résultat

Le sujet peut faire référence à une action ou à son résultat pour plusieurs raisons :

- Il peut demander de refaire une nouvelle action du même type (requête recommencer)
- Il peut annuler une action qui vient d'être réalisée.
- Il peut faire directement référence au résultat d'une action lorsqu'il juge le résultat visible.

Dans le Tableau 23, « le résultat » fait référence au résultat de l'action « faire un zoom avant ». Le codage adopté correspond à une balise `<rs>` de type objet car à la fin de l'action, ce qui est visible est le même objet mais vu différemment.

<u id= "u27" who= "S2"> **Faire un zoom avant**
<u id= "u28" who= "C">Bien reçu. Opération effectuée. **Le résultat** vous convient- il ?
<u id= "u29" who="S2">Oui.</u>

⇓

<u id= "u27" who= "S2"><rs type= "action" key= "A1" >Faire un zoom avant<\rs>.<\u>
<u id= "u28" who= "C">
<seg id= "u28seg1"> Bien reçu.
<seg id= "u28seg2"> Opération effectuée.
<seg id= "u28seg3"> <rs type= "objet " key= "R1">Le résultat<\rs>vous convient- il ?<\u>
<u id= "u29" who="S2">Oui.</u>

Tableau 23 : Codage d'une référence à une action

4.2 Référence à un ensemble d'actions

En général, le sujet ne fait pas référence à une seule action mais à un groupe d'actions. Il est dès lors intéressant de pouvoir délimiter les actions à prendre en compte suite à l'utilisation de verbes tels que recommencer (Tableau 24).

<u id="u27" who="C">La surface est contrainte. Désirez-vous que je l'interpole?</u>
<u id="u28" who="S1">Oui</u>
<u id="u29" who="C">Opération en cours. Opération terminée. Le résultat vous convient-il?<\u>
<u id="u30" who="S1">Est-ce que tous les endroits où il n'y avait pas correspondance ont été affinés ?</u>
<u id="u31" who="C">Le maillage de la surface est trop grossier par rapport à la quantité de points</u>
<u id="u32" who="S1">Peut-on alors **recommencer** l'opération avec un maillage plus fin ?</u>

Tableau 24: Référence à un groupe d'actions

Dans le tour de parole u32, « recommencer l'opération » est à mettre en relation directe avec « Le maillage de la surface est trop grossier par rapport à la quantité de points »(u31).

On ne sait pas quelle action ou quel groupe d'actions il faut recommencer. Faut-il interpoler à nouveau ou bien remonter plus haut dans le dialogue? Dans ce cas, la proposition est de structurer les actions en groupes d'actions comme dans le Tableau 25 et de cette manière il pourra être fait référence au dernier groupe d'actions.

```
<u id="u27" who="C">La surface est contrainte. Désirez-vous que
        <rs type="action" key="A1"> je l'interpole<\rs> ?</u>
<u id="u28" who="S1">Oui</u>
<u id="u29" who="C">Opération en cours. Opération terminée.
        <rs type="objet" key="O1" target="A1">
        Le résultat</rs> vous convient-il?<\u>
<u id="u30" who="S1">Est-ce que tous les endroits où il n'y avait pas
        correspondance ont été affinés ?</u>
<u id="u31" who="C">Le maillage de la surface est trop grossier par
        rapport à la quantité de points</u>
<u id="u32" who="S1">Peut-on alors
        <rs type="action" key="A2">recommencer l'opération</rs>
        avec un maillage plus fin ?</u>

<link type="relation" target="A2 A1"></link>
```

Tableau 25: Structuration des actions

5 Autres références

Les références ne sont pas uniquement associées à des actions, mais également à des individus ou à des propriétés.

5.1 A un individu

Dans le corpus GOCAD, les dialogues sont réalisés entre deux interlocuteurs ; il est donc facile de noter les références à l'un et à l'autre. Un pronom « vous » désignera nécessairement le sujet s'il apparaît dans un énoncé du compère et inversement. Cependant, il peut arriver que les sujets dialoguent avec la machine comme s'il s'agissait de dialogues homme-homme.

Les sujets utilisent également plusieurs formes verbales pour s'adresser à la machine telles que l'impératif ou encore l'infinitif (Tableau 26).

```
sujet: Créer des surfaces... Ah pourquoi il change de vue?
sujet: Fais une rotation
sujet: Créer la surface
```

Tableau 26: Formes verbales utilisées pour s'adresser à la machine

Un certain nombre de problèmes se posent pour coder les références aux individus :

- Faut-il uniquement coder les références explicites ou faut-il aussi coder les références implicites, et donc aller au-delà des informations fournies par le texte.
- Trouver l'interlocuteur est facile lorsqu'il n'y a que deux individus, mais le problème se complexifie lorsque le nombre d'individus augmente. En effet, comment noter dans un trilogue le fait que le locuteur fait référence à un individu particulier ou aux deux.

5.2 A une propriété

Un même syntagme peut à la fois correspondre à un objet ou à une propriété.

Ainsi il faut faire la différence entre:« Changer la **couleur** de la surface. » (*La couleur est une propriété de l'objet surface.*) et « Quelles sont les **couleurs** disponibles? » (*Les couleurs correspondent à un objet spécifique.)*

Le Tableau 27 propose une solution pour le codage de la référence à une propriété.

```
<rs type="propriété" key="P1">la couleur de<rs type="objet" key="O1">
la surface<\rs><\rs>
```

Tableau 27: codage d'une référence à une propriété

6 Un univers de référence

Il est possible de faire référence à un objet même si celui-ci n'est pas concrètement présent dans l'espace où se trouve le locuteur, donc, il est difficile de dire si la référence se réalise effectivement. Il est dès lors nécessaire de pouvoir indiquer parmi quel ensemble d'objets le locuteur réfère.

Une proposition pour décrire les objets accessibles à un moment donné est de faire appel à la balise <univers> comme dans le Tableau 28. Elle répertorie l'ensemble des objets parmi lesquels des références pourraient être faites.

```
<u id="u3" who='S2">Charger le fichier Vtop1 point ts
<u id="u4" who="C">Opération effectuée

                    ⇓

<univers type="objet">le fichier Vtop1 point ts</univers>
        <u id="u3" who="S2">Charger <rs type="objet" key="F1">le fichier
                Vtop1 point ts </rs> </u>
        <u id="u4" who="C">Opération effectuée</u>
```

Tableau 28: Définition d'un univers

En général, il ne sera pas nécessaire de posséder une liste totale des objets présents, les références se feront plutôt au niveau des sous-univers.

Dans le Tableau 29, un certain nombre d'objets se trouvaient à l'écran au début de la tâche, puis le sujet a créé trois surfaces particulières. Ces trois surfaces peuvent être considérées comme appartenant à un sous-univers.

La balise `<link type=SourceUnivers>` permet de définir dans quel univers les trois objets devront être recherchés. Par la suite, les références ultérieures pourront se faire soit directement à la surface vue dans son ensemble (l'objet "Ok") soit au trois surfaces (les objets O1, O2 et O3) qui ont permis la création de "Ok".

```
<u id="u33" who="S5">Densifier le maillage
<u id="u34" who="C">Bien compris. Quelle surface désirez-vous
        densifier ?
<u id="u35" who="S5">Les 3
<u id="u36" who="C">Voulez-vous relier vos 3 surfaces pour n'en
        former qu'une seule</rs> ?</u>
<u id="u37" who="55">Oui
<u id="u38" who="C">Bien compris. Quel nom voulez-vous donner à
        votre  surface?
<u id="u39" who="S5">Dome

                                ⇓

<univers type="objet" key="U1">surfaces O1 O2 O3</univers>

<u id="u33" who="S5">Densifier le maillage</u>
<u id="u34" who="C">Bien compris. <rs type="objet" key="Ox">Quelle
        surface</rs> désirez-vous densifier ? </u>
<u id="u35" who="S5"><rs type="objet" key="Oy">Les 3</rs></u>

<link type="SourceUnivers" targets="Oy, U1"></link>

<u id="u36" who="C">Voulez-vous relier <rs type="objet" target="Oz">
        vos 3 surfaces</rs> pour n'en former qu' <rs type="objet"
        key="Ok"> une seule</rs> ?</u>
<u id="u37" who="55">Oui</u>
<u id="u38" who="C">Bien compris. Quel nom voulez-vous donner à
        <rs type="objet" target="Ot">votre surface</rs> ? </u>
<u id="u39" who="S5"><rs type="objet" key="Os">Dome</rs></u>
```

Tableau 29 : Définition d'un sous-univers

Un des grands avantages de définir un univers pour la référence, c'est qu'il permettra de résoudre des coréférences plus complexes. Dans le Tableau 30, sujet et compère désignent différemment un même objet. On pourra savoir si la coréférence est possible en prenant en compte le contenu de l'univers. Si le marquage de la coréférence n'est pas certain, l'univers doit au moins permettre de fournir les candidats les plus probables.

<u id="u5" who="S1">Peut-on changer les couleurs, mettre du jaune à la place du noir et du bleu aussi ?
<u id="u6" who "C">Vous voulez changer la couleur des deux surfaces?
<u id="u7" who="S1">Oui mettre **la surface noire** en jaune et non garder **la surface bleue** en bleu
<u id="u8" who="C">Opération effectuée
<u id="u13" who="S1">Peut-on tracer les points d'intersection entre **la surface H2** et **la surface H3**

⇓

<u id="u5" who="S1">Peut-on changer les couleurs, mettre du jaune à la place du noir et du bleu aussi ? </u>
<u id="u6" who "C">Vous voulez changer la couleur des deux surfaces?</u>

<univers type="objets" key="U1">O1 O2</univers>

<u id="u7" who='S1">Oui mettre <rs type="objet" key="Ow">la surface noire</rs>en jaune et non garder <rs type="objet" key="Ox">la surface bleue</rs>en bleu</u>

<link type="SourceUnivers" targets="(Ox, Ow), U1"></link>

<u id="u8" who="C">Opération effectuée</u>
<u id="u13" who="S1">Peut-on tracer les points d'intersection entre <rs type="objet" key="Oy">la surface H2</rs>et <rs type="objet" key="Oz">la surface H3</rs>?</u>

Tableau 30 : codage de coréférences complexes

Au début du dialogue, le sujet et le compère font référence à deux surfaces qu'ils désignent d'abord par leur couleur (la surface noire et la surface bleue) puis par leur nom (la surface H2 et la surface H3). La balise <univers> permet de dire qu'il y a lien entre les objets nommés par leur couleur et ceux nommés par leur nom mais ne permet pas nécessairement d'expliciter ce lien.

L'univers est défini par les actions qui ont été réalisées précédemment. La balise <link> montre le lien qui existe entre les deux objets (Ox et Ow) et l'univers U1. C'est donc dans l'univers U1 qu'il faut rechercher la source des deux expressions référentielles.

Remarques:

- Un objet est considéré comme faisant partie d'un univers qu'à partir du moment où il a été explicitement nommé ou désigné par un geste.

- Par analogie, on peut définir un univers d'actions qui recensera les actions effectuées.
- La définition d'un univers des personnes peut s'avérer intéressant suivant le type de document traité. Dans la cas de GOCAD, cet univers est constant et ne doit être décrit qu'une seule fois. Par contre pour une pièce de théâtre, il serait intéressant de définir l'univers des personnages présents dans chaque chapitre de manière à éviter d'introduire dès le début des personnages qui n'interviennent que bien plus tard ou encore de garder dans l'univers de départ des personnes qui ne sont plus sur scène.

7 Etablir des liens entre des syntagmes référentiels

Il n'est pas toujours évident de dire si deux SN coréfèrent à des objets déjà apparus dans l'énoncé. Dés lors , dans le Tableau 31, les référence et coréférence sont séparées dans le codage. Les coréférences seront indiquées par une balise <link>.

```
<u id="u11" who="S2"> Construire la surface
<u id="u12" who="C"> Opération effectuée.
<u id="u17" who="52">Modéliser la surface
<u id="'u22" who="C"> Quel nom désirez-vous donner à votre
      surface?
<u id="u23" who="S2">X1
<u id="u24" who="C"> Bien compris. Opération en cours. Opération
      effectuée.
                                    ⇓
<u id="u11" who="S2"> Construire <rs type="objet" key="O1">la
      surface</rs> </u>
<u id="u12" who="C"> Opération effectuée. </u>
<u id="u17" who="52">Modéliser <rs type="objet" key="O2">la
      surface</rs>  </u>
<link type="coref" targets="O2 O1"></link>
<u id="'u22" who="C"> Quel nom désirez-vous donner à <rs
      type="objet"  key="O3">votre surface</rs> ? </u>
<link type="coref" targets="O3 O2"></link>
<u id="u23" who="S2">X1</u>
<u id="u24" who="C"> Bien compris. Opération en cours. Opération
      effectuée.</u>
```

Tableau 31 : Utilisation de la balise <link> pour coder les coréférences

La balise <link> met en relation une expression référentielle avec la dernière expression référentielle précédente référant au même objet. L'attribut target permet d'indiquer entre quels objets le lien est réalisé.

7.1 Etude du syntagme [N1 de N2]

Le but de ce point est de montrer l'hétéréogénéité des relations à prendre en compte si l'on veut réaliser un codage complet des relations entre objets.

7.1.1 [N1 de N2]: expressions figées

Dans le cas du corpus GOCAD, on pourrait imaginer la création d'une liste reprenant les expressions figées comme « point de vue », « plan de coupe ». Dans d'autres types de documents, cela s'avérerait difficile.

7.1.2 Différentes valeurs pour « de N2 »

Lorsqu'il n'y pas de détermination devant N2, « de N2 » à une valeur adjectivale qui correspond à une caractérisation de l'objet désigné par N1. Les différentes valeurs pour « de N2 » sont décrites dans le Tableau 32.

Valeur pour « de N2 »	Exemples
Valeur locative	• Spatiale: la surface **de droite** • Temporelle la surface **de base** (par opposition aux autres surfaces ayant subi des modifications)
valeur de mesure	une rotation **de 40 degrés**
valeur de fonction	les points **d'intersection**

Tableau 32 : Différentes valeurs pour « de N2 »

7.1.3 N1 comme nom de propriété

Lorsque N1 correspond à un nom de propriété, on peut montrer l'équivalence entre [N1 de N2] et [N2 être 'adjectif'] comme dans l'exemple suivant:

« le bleu de la surface = la surface est bleue »

7.1.4 N2 comme objet de référence

Lorsque N2 est déterminé, le syntagme « N1 de (déterminant) N2 » indique une référence à un objet N1qui est rendue possible par le fait que N2 est déjà identifié.

7.2 Typologie des liens

Suite à l'étude du corpus GOCAD, deux types de liens ont pu être dégagés, les liens internes et externes. Les liens internes sont ceux où les objets reliés sont explicitement marqués par des balises <rs> dans le texte. Ils vont être codés à l'aide de la balise <link>. Les liens externes, quant à eux, permettent de faire le lien entre des syntagmes référentiels du texte et des gestes de désignation.

7.2.1 Liens internes

Les liens internes abordés sont la coréférence, le partitif, la localisation, l'inclusion, la possession, la composition, l'association et la propriété.

- La coréférence

Il y a coréférence lorsque deux expressions référentielles (<rs>) désignent le même objet du monde. Dans le Tableau 33, les deux syntagmes « Les surfaces S1 et S2 » et « Les » désignent le même objet. La balise <link> indique le type de relation liant les deux syntagmes.

```
<u id= "u145" who= "C">Les surfaces S1 et S2  ne sont pas
        visualisées à l'écran.
<u id= "u146" who= "S7"> Les visualiser.

                                ⇓

<u id= "u145" who= "C"> <rs type= "objet" key= "O1">Les surfaces S1
        et S2 </rs> ne sont pas visualisées à l'écran.</u>
<u id= "u146" who= "S7"> <rs type= "objet"  key= "O2">
        Les</rs>visualiser.</u>
<link type= "coref " relation= "interne" target= "O2 O1"
        targetOrder= "y"></link>
```

Tableau 33 : Codage de la coréférence

- Le partitif

Il s'agit de marquer les relations « partie-tout ». Dans le Tableau 34, le premier élément de l'attribut target correspond à la partie et le second au tout.

Paul aime sa **voiture** parce que les **sièges** sont confortables.

⇓

Paul aime <rs type="objet" key="O1">sa voiture</rs>parce que <rs type="objet" key="O2">les sièges</rs> sont confortables.

<link type="partitif" relation="interne" target="O2 O1" targetOrder="y></link>

Tableau 34 : Codage du partitif

- La localisation

Il y a localisation lorsqu'un objet est identifié dans l'espace par rapport à un autre (Tableau 35).

Supprimer la partie de la surface S se trouvant au dessus de la surface S1.

⇓

Supprimer <rs type="objet" key="O1">la partie de la surface S</rs> se trouvant au dessus de <rs type="objet" key="O2">la surface S1</rs>.

<link type="localisation" relation="interne" target="O1 O2" targetOrder="y"></link>

Tableau 35 : Codage pour la localisation

- L'inclusion

Il y a inclusion lorsqu'un objet est entièrement situé à l'intérieur d'un autre (Tableau 36).

Créer une cage autour de la surface S1.

⇓

Créer <rs type="objet" key="O1">une cage</rs> autour de <rs type="objet" key="O2">la surface S1</rs>.

<link type="inclusion" relation="interne" target="O2 O1" targetOrder="y"></link>

Tableau 36 : Codage pour l'inclusion

- La possession

Cette relation exprime le fait qu'un objet appartienne à un individu (Tableau 37).

Le chien de Sophie.

⇓

```
<rs type="objet" key="O1">Le chien</rs> de <rs type="personne"
      key="O2">Sophie</rs>.

<link  type="possession"  relation="interne"  target="O2 O1"
      targetOrder="y"  </link>
```

Tableau 37 : Codage pour la possession

- La composition

Cette relation indique qu'un objet est considéré comme entièrement composé d'objets d'un autre type (Tableau 38).

les arbres de la forêt

⇓

```
<rs type="objet" key="O1">les arbres</rs> de <rs type="objet"
      key="O1">la forêt</rs>

<link  type="composition"  relation="interne"  target="O1 O2"
      targetOrder="y"></link>
```

Tableau 38 : Codage pour la composition

- L'association

Cette relation indique ce qui est généralement désigné par anaphore associative.

Il y avait une tasse sur la table.
L'anse était cassé.

⇓

Il y avait <rs type="objet" key="O1">une tasse</rs> sur la table.
<rs type="objet" key="O2">L'anse</rs> était cassé.

<link type="association" relation="interne" target="O2 O1"
targetOrder="y"></link>

Tableau 39 : Codage pour l'association

- La propriété

Une expression référentielle désigne une propriété d'un objet X lorsque l'on peut la reformuler de la manière suivante : « X est Y » avec Y comme attribut. Ainsi, dans le Tableau 40, l'expression « le courage de Socrate » devient « Socrate est courageux » avec « courageux » comme attribut.

le courage de Socrate

⇓

<rs type="propriété" key="O1">le courage</rs> de
<rs type="personne" key="I1">Socrate</rs>

<link type="propriété" relation="interne" target="O1 I1"
targetOrder="y"></link>

Tableau 40 : Codage pour la propriété

7.2.2 Liens externes

Il peut arriver que le sujet désigne des objets de l'écran par gestes uniquement ou en combinant la parole et le geste.

Ainsi, l'expression « Appeler cette surface S1 » devra être analysée différemment selon qu'elle sera ou non accompagnée d'un geste.

- Une expression référentielle accompagnée d'un geste

Comme on peut le voir dans le Tableau 41, une ancre (la balise <anchor>) permet de synchroniser le geste et la parole.

<u id="u41" who="S4">Enlève **cette partie là(+désignation)**
<u id="u42" who="C">Message reçu. Veuillez patienter.
<u id="u43" who="S4">O.K.

⇓

<u id="u41" who="S4">Enlève <rs type="objet" key="O1>cette partie là
</rs><anchor id="a1" synchro="K1"></anchor>
<kinesic id="K1" desc="click sur sur le diapir"></kinesic>
</u>

<link type="démonstration" relation="externe" target="O1 K1"
targetOrder="y"></link>

<u id="u42" who="C">Message reçu. Veuillez patienter.</u>
<u id="u43" who="S4">O.K.</u>

Tableau 41 : Codage pour le geste

- Une expression référentielle accompagnée de plusieurs gestes

Il s'agit du même principe de codage que pour un geste unique. Le codage est un peu plus complexe, comme on peut le constater dans le Tableau 42, étant donné le nombre d'éléments à prendre en compte,.

<u id="u16" who="C">Je ne peux construire de surface qu'entre 2
courbes
<u id='u17" who="S1">Peut-on alors tracer la surface</rs> entre **la ligne supérieure** et **la ligne intermédiaire** du dôme?
<u id="u18" who="C">Pouvez-vous désigner **ces lignes**
<u id="u19" who="S1">**(désignation)(désignation)**
<u id="u20" who="C">Message reçu. Opération en cours.

⇓

<u id="u16" who="C">Je ne peux construire de surface qu'entre 2
courbes</u>
<u id="u17" who="S1">Peut-on alors tracer <rs type="objet"
key="O1">la surface</rs> entre <rs type="objet" key="O2">
<rs type="objet" key="O3">la ligne supérieure</rs> et
<rs type="objet" key="O4">la ligne intermédiaire</rs></rs> du
<rs type="objet" key="O5">dôme</rs></rs> ?
</u>

```
<u id="u18" who="C">Pouvez-vous désigner <rs type="objet"
        key="O6">ces lignes</rs> ?</u>

<u id="u19" who="S1">
 <anchor id="a2" synchro="K2"></anchor>
 <anchor id="a1" synchro="K1"></anchor>
 <kinesic id="K2" desc="click sur courbeduhaut"></kinesic>
 <kinesic id="K1" desc="click sur courbeendessous"></kinesic>
</u>

<u id="u20" who="C">Message reçu. Opération en cours.</u>

<link type="démonstration" relation="externe" target="O3 K2"
        targetOrder="y"></link>
<link type="démonstration" relation="externe" target="O4 K1"
        targetOrder="y"></link>
```

Tableau 42 : Codage pour plusieurs gestes