Académie de Montpellier
Université Montpellier II
Sciences et Techniques du Languedoc

MÉMOIRE DE

STAGE RECHERCHE DE MASTER M2 INFORMATIQUE

effectué au Laboratoire d'Informatique de Robotique
et de Micro-électronique de Montpellier

Spécialité : **AIGLE**

Inférence de la structure d'une page web en vue d'améliorer son accessibilité

par **Franck PETITDEMANGE**

Date : **27 juin 2014**

Sous la direction de **Marianne HUCHARD, Michel MEYNARD et Yoann BONAVERO**

Remerciements

Je remercie mes encadrants Marianne Huchard, Michel Meynard et Yoann Bonavero pour leur patience, leur disponibilité ainsi que leurs précieux conseils qui ont guidé ma démarche dans ce travail de recherche.
Je remercie également les membres du jury et en particulier mes rapporteurs Rodolphe Giroudeau, Mathieu Lafourcade et Philippe Vismara pour le temps accordé à la lecture du rapport et dont les remarques éclairées me seront utiles.

Table des matières

Table des figures

Glossaire

Document Object Model Le Document Object Model (ou DOM) est un standard du W3C qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents XML et HTML *wikipedia*. 23

Ontologie En philosophie, l'ontologie est l'étude de l'être en tant qu'être, c'est-à-dire l'étude des propriétés générales de ce qui existe. Par analogie, le terme est repris en informatique et en science de l'information, où une ontologie est l'ensemble structuré des termes et concepts représentant le sens d'un champ d'informations *wikipedia*. 17

Pattern Le mot anglais pattern est utilisé pour désigner un modèle, une structure, un motif. 12

W3C Le World Wide Web Consortium, abrégé par le sigle W3C, est un organisme de normalisation à but non lucratif, fondé en octobre 1994 chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML. 12

Chapitre 1

Introduction

Le World Wide Web (www) est un réseau de ressources. Leur publication repose sur un langage universellement compréhensible et accepté par tous les terminaux : HTML. Historiquement conçu pour faciliter l'échange d'articles dans la communauté scientifique, la démocratisation du web a fait radicalement évoluer le contenu des pages web, sans pour autant que le langage ne suive cette évolution. Le contenu des pages web est passé d'un contenu homogène, une page contenant un simple texte structuré, à un contenu hétérogène. En effet, de nos jours, les pages sont constituées de différentes structures organisant différentes informations. Les différentes structures peuvent être la bannière de la page qui renseigne sur l'identité du site, un menu de navigation qui présente l'organisation du site web ou un article qui caractérise l'information principale. La version 4 de HTML est le langage de publication le plus utilisé actuellement. Ce langage est complètement inadapté à la description d'un contenu hétérogène. Le langage ne permet pas de signaler différents contenus. Il n'y a aucune sémantique associée aux éléments de langage pour décrire les structures qui organisent les différentes informations dans une page : on ne peut pas annoter une structure comme étant un menu de navigation de manière formelle et standardisée. De plus HTML 4 possède une composante syntaxique extrêmement souple permettant la construction d'un nombre potentiellement infini de constructions pour décrire le contenu d'une page.

Ce manque de compréhension de la structuration d'une page complexifie la compréhension, par un processus automatique, du contenu d'une page. Ce manque de compréhension du contenu des pages impacte par exemple les processus d'indexation des moteurs de recherche. Dans le contexte du stage on s'intéresse particulièrement à la compréhension du contenu pour les outils d'accessibilité destinés aux personnes en situation de handicap visuel.

La question de recherche posée par le stage s'intéresse au moyen d'inférer, par un processus automatique, les différentes informations structurant une page web. On peut identifier deux étapes dans ce processus d'inférence. La première étape consiste à isoler les différentes structures logiques dans une page. Nous employons le terme de structures logiques pour désigner un ensemble d'éléments qui mis en relation remplissent une fonction (*e.g* un menu, une barre de recherche). La seconde étape consiste à comprendre la fonction des structures isolées. Cette question d'inférence intéresse des travaux de thèse en cours de réalisation sur l'accessibilité du web [1] [2]. Une des contributions visées par cette thèse est d'adapter le contenu d'une page aux déficiences visuelles partielles d'un utilisateur. C'est ce processus d'adaptation qui implique le besoin de comprendre le contenu d'une page.

Pour aborder cette problématique, nous étudions dans un premier temps l'environnement des pages web afin de conceptualiser les différents éléments constitutifs de celles-ci et de se détacher d'un langage en particulier. Cette étude se concrétise par la réalisation d'un méta-modèle de page web. Nous validerons ce méta-modèle en évaluant son expressivité par la modélisation des préférences d'utilisateurs sur l'accessibilité des pages web. Une fois cette étape achevée nous élaborons une méthode d'extraction des différents éléments constitutifs d'une page web. Cette méthode sera complétée par un processus d'annotation de ces éléments d'après notre méta-modèle. La validation de l'approche consiste à vérifier la conformité des éléments extraits et leurs annotations d'après les spécifications du méta-modèle construit.

La problématique provient principalement du langage HTML 4. Comme expliqué plus haut HTML 4 ne possède pas une sémantique adaptée à la description de son contenu. A cela s'ajoute une grande souplesse syntaxique qui amène deux difficultés. La première est une diversité de structure dans la représentation des informations d'une page (*e.g* on peut représenter un menu de dix façons différentes). Les structures ne sont pas conformes à un modèle de représentation. Ce premier point écarte une approche basée sur la reconnaissance de Pattern. Le second problème de la syntaxe souple est qu'il est possible de construire une page qui ne respecte pas les spécifications de la W3C. Ce point implique que l'on ne peut pas effectuer un découpage du DOM, pour extraire les différents contenus, en respectant seulement la sémantique associée aux étiquettes HTML (d'après la norme du W3C). D'autre part, la construction des pages web repose également sur un langage pour décrire la mise en forme. Le problème est un possible écart entre la structure du DOM et la structure de la page au moment de l'affichage dans le navigateur.

Pour définir notre méthodologie, le constat, qui oriente la direction de nos travaux, est que la structure d'une page est explicitée au moment de la mise en page (par le navigateur), c'est-à-dire par l'application des différents styles de mise en forme

associés aux balises HTML. Nous pensons donc qu'il est possible d'exploiter la mise en forme associée aux éléments de langage pour isoler les groupes qui structurent une information. Notre approche se base sur un algorithme existant que nous améliorons pour que ce dernier prenne mieux en compte la conception actuelle des pages web.

Ce rapport se structure comme suit. Le chapitre 2 présente des préférences d'utilisateurs sur l'accessibilité. Ces préférences sont extraites de travaux réalisés antérieurement à ce stage. Le chapitre 3 présente un état de l'art. Il introduit tout d'abord une synthèse des langages de publication que nous avons réalisée à partir des spécifications des différents standards. Nous y présentons également des approches existantes permettant d'isoler les différentes structures logiques ainsi que des solutions pouvant leur associer une sémantique. Dans le chapitre 4 nous présentons la contribution de notre stage. Dans la première partie du chapitre nous proposons un méta-modèle de page web créé. Dans la seconde partie du chapitre, nous proposons l'adaptation d'une méthode de la littérature pour identifier la structure d'une page. Cette approche fonctionne sur la base d'un algorithme de segmentation basé sur des heuristiques que nous avons développées. Cette méthode est complétée par un processus d'annotation des structures extraites de la phase précédente. Le chapitre 5 présente et analyse les premiers résultats et une discussion de ceux-ci. Pour conclure le chapitre 6 présente une synthèse des travaux effectués et leurs perspectives.

Chapitre 2

Accessibilité et contexte du stage

Ce stage s'inscrit dans le contexte d'une thèse en cours de réalisation sur l'accessibilité numérique [2]. L'un des objectifs de la thèse est prendre en compte les préférences d'un utilisateur en situation de déficience visuel par un processus d'adaptation de pages web automatisée. L'accessibilité désigne le caractère possible de la liberté de déplacement dans l'espace, d'utilisation d'outils, et de compréhension. Appliqué aux pages web l'accessibilité signifie la capacité d'accéder aux informations contenues dans une page et d'interagir avec. Certains handicaps peuvent brider l'accessibilité. Les handicaps peuvent être physiques, sensoriels (visuels, auditifs...), cognitifs, mentales ou psychiques. La thèse se concentre sur les handicaps visuelles. Le constat de l'état de l'art réalisé (dans la thèse) met en évidence que les outils existants pour améliorer l'accessibilité d'une page ne prennent pas en compte le profil des utilisateurs. Les outils sont peu personnalisables et possèdent des fonctionnalités d'accessibilité très génériques. Ce point pose problème car certains types d'adaptation qui rendent accessible un contenu pour une personne peut rendre ce même contenu inaccessible pour d'autres personnes. De plus, certaines adaptations résolvant le problème sur une partie d'une page peuvent créer des difficultés ailleurs. Cette configuration implique le besoin de prendre en compte les préférences d'un utilisateur.

La figure 2.1 présente le processus global d'adaptation de pages web. Ce processus d'adaptation est une proposition faite dans ce stage pour faciliter le traitement des préférences d'un utilisateur. Il s'agit d'un processus d'ingénierie dirigée par les modèles. L'idée est de s'intéresser à la conception des pages d'un point plus abstrait ce qui offre une solution au problème du traitement des différents formats de données du web (diversité de représentation des données). D'autre part, on ne manipule que les concepts qui nous intéressent pour modéliser les préférences utilisateurs. Le schéma

présente des pages sources (pages web) conformes à un méta-modèle de HTML 4. Ce méta-modèle n'est pas un support adapté à l'expression de préférences, nous développons ce point de vu dans la section 4.1. Le schéma décrit donc une première transformation d'une page source vers un modèle source conforme à un méta-modèle intermédiaire plus adapté à la modélisation de préférences. Une fois cette représentation plus abstraite, construite une seconde étape de transformation intervient pour appliquer les préférences des utilisateurs. Une fois ces préférences appliquées, une troisième transformation intervient pour générer une nouvelle page conforme aux préférences de personnalisation de l'utilisateur dans les outils d'accessibilité. Par rapport à ce schéma, l contribution de ce stage se situe dans la réalisation du méta-modèle intermédiaire et dans l'extraction des éléments du méta-modèle intermédiaire dans une page web conforme à HTML 4.

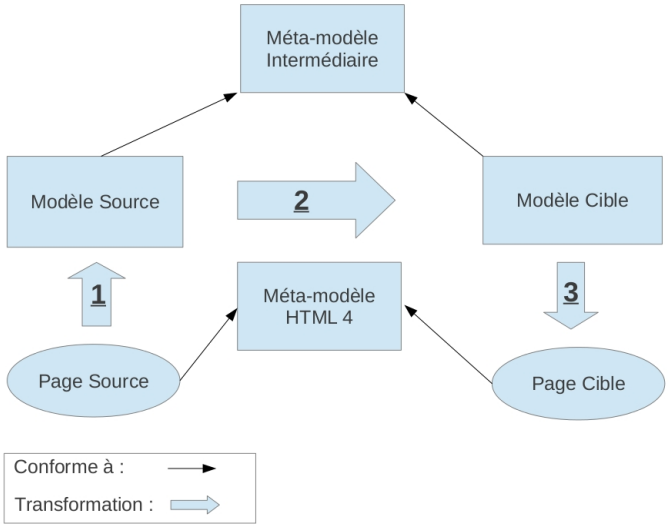


FIGURE 2.1 – Processus d'adaptation d'une page web

Ci-dessous nous présentons plusieurs types (non-exhaustives) de problèmes redondants dans l'accessibilité des pages web :

problèmes de surcharge visuelle : des pathologies impliquant une base vision qui nécessitent l'utilisation de fort zoom sur une page. Si la page web contient trop d'informations, la navigation dans le site devient difficile. En effet, le lecteur n'a accès qu'à un fragment de la page. La réduction de la vision globale implique un effort de concentration pour se représenter le contexte général afin

de se situer. Si la page est très fournie en information il doit mémoriser et positionner mentalement beaucoup d'éléments à chaque déplacements.

problèmes de polices de caractère : les personnes âgées sont sujettes à des dégénérescences de la rétine ce qui entraîne une diminution de la précision de la vue. Ces catégories de personne deviennent plus sensibles à la taille des polices de caractère. Pour rendre accessible le texte, les outils d'accessibilité augmentent la taille des caractères. La trop grande différence de taille entre les caractères du corps d'un texte et les éléments de titre les rendent alors illisibles.

problèmes de contrastes : des pathologies comme le daltonisme entraînent une distorsion dans la perception des couleurs et potentiellement un contraste plus faible que celui d'origine. Des couleurs trop proches entre un texte et son arrière plan peuvent rendre ce texte illisible.

D'après ces quelques éléments on peut en déduire certaines préférences qui par leur modélisation valideront notre méta-modèle. Par exemple les problèmes de surcharge visuelle peuvent être pris en compte en masquant les éléments de la page qui n'intéressent pas le lecteur (*e.g* bannière, pied de page, menu). Les problèmes de police qui dans l'exemple donné vont être exprimés par une contrainte de différence de taille maximale entre les éléments de titre et le corps du texte qu'il introduit. Les contrastes seront traités par une restriction sur certaines combinaisons de couleur entre un texte (avant-plan) et son arrière plan.

Chapitre 3

État de l'art

L'état de l'art s'organise autour d'une synthèse des langages de publication, il souligne les principaux concepts de ceux-ci. Puis nous présentons quelques articles traitant du partitionnement du DOM des pages web qui nous intéresse particulièrement pour l'extraction des structures logiques. Pour finir nous évoquons une approche permettant d'attribuer une sémantique aux structures composant une page web.

3.1 Langage de publication de page web

3.1.1 HTML 4

HTML 4 [5] est un langage permettant la publication de contenu sur le web. C'est le langage standard actuel des pages web. Il permet de structurer le contenu et de lui associer une mise en forme. Le contenu est un contenu multimédia (texte, images, *etc.*). Ce contenu est organisé de manière hiérarchique en le découpant en sections et sous-sections.

Contenu Le contenu principal décrit dans les pages HTML 4 est un contenu textuel. Il peut également contenir du multimédia comme des images, des vidéos et applets (des programmes qui sont automatiquement chargés puis lancés sur la machine de l'utilisateur). L'inclusion de contenu multimédia se fait par l'élément générique : <object>. Il possède une collection d'attributs prédéfinis qui décrivent l'objet inclus dans la page. Le principal étant *type* décrivant le type de contenu des données (*e.g.* figure 3.1). La valeur de ces attributs n'est pas prédéfinie. Elle est interprétée librement par la machine qui charge la page web.

```
<object data="data/test.mpg" type="video/mpeg">
        Ceci est une vidéo
</object>
```

FIGURE 3.1 – Exemple contenu multimédia

Structuration générique HTML 4 propose un mécanisme générique pour la composition de contenu formant la structure des pages web. Ce mécanisme gravite autour des éléments de type <DIV> et de leurs attributs respectifs : *id* et *class*.

DIV Signifiant division, la balise DIV est utilisée comme conteneur générique, il peut contenir n'importe quel élément. Il est exploité pour :

- regrouper les éléments pour leur appliquer un style (une mise en forme particulière).
- signaler une section ou une sous-section.

id et class Chaque élément peut se voir attribuer un identifiant ou une classe d'appartenance. *id* assigne un nom à un élément. Ce nom est unique dans le document. *class* au contraire, assigne un ou plusieurs noms de classe à un élément. Un nom de classe peut être partagé par plusieurs instances d'éléments. Les identifiants et les classes sont des suites de caractères quelconques décidées arbitrairement par l'auteur du document.

Les éléments DIV utilisés conjointement avec les attributs *id* et *class* sont au cœur du mécanisme générique de structuration d'un document. DIV permet de diviser le contenu d'un document en sections et sous-sections (*e.g.* figure 3.3) pour décrire sa structure. Les balises ayant une sémantique neutre, c'est l'auteur du contenu qui attribue (de manière arbitraire) un nom de *class* ou un *id*. L'*id* ou la *class* est associé à une mise en forme définie a priori. La mise en forme est définie au travers d'un langage : Cascading Style Sheets (CSS) [6] que l'on appelle feuille de style. CSS permet d'appliquer un ensemble de règles de style ou un agencement des éléments dans l'espace de la page. Par exemple, l'auteur peut déclarer une classe "aside" et définir que les éléments appartenant à la classe "aside" doivent être placés sur le côté droit de la page avec un fond blanc. Ce mécanisme est illustré par la figure 3.2. L'auteur associe à chaque <DIV> une *class* ou un *id* auquel s'applique une mise en page et une mise en forme définies par l'auteur dans une feuille de style CSS.

```
<body>
        <div id="header" ></div>
        <div id="navigation_bar"></div>
        <div class="aside"></div>
        <div class="section">
                <div class="article"></div>
                <div class="article"></div>
        </div>
        <div class="aside"></div>
        <div id="footer"></div>
</body>
```

FIGURE 3.2 – Architecture page web HTML 4

3.1.2 HTML 5

HTML 5 [8] étend HTML 4 en apportant de nouveaux éléments lexicaux. Ces nouveaux éléments apportent une sémantique standard et de plus haut niveau. Elle permet notamment d'expliciter la structure d'une page.

Contenu HTML 5 fournit de nouveaux éléments comme <video>, <audio> avec un ensemble d'attributs propres à chaque balise (a contrario de l'élément <object> de HTML 4). Les attributs spécifiques permettent de renseigner l'état d'un élément. Par exemple, la balise <audio> possède un attribut spécifique *muted* indiquant si le

```
<body>
<div class="section" id="elephants-foret" >
        <h1>Les éléphants des forêts</h1>
        <p>Dans cette partie, nous abordons le sujet
moins connu des éléphants des forêts.</p>
        <div class="sous-section" id="habitat-foret" >
        <h2>L'habitat</h2>
        <p>Les éléphants des forêts ne vivent pas
dans les arbres mais au milieu d'eux.</p>
        </div>
</div>
</body>
```

FIGURE 3.3 – Exemple découpage en sections et sous-sections

son de l'élément audio est coupé ou non.

Structuration Les nouveaux éléments de HTML 5 spécifient donc une sémantique standard. Ci-dessous la liste des principaux éléments :

- SECTION : représente une section générique dans un document, c'est-à-dire un regroupement de contenu par thématique.
- ARTICLE : représente un contenu autonome dans une page, facilite l'inclusion de plusieurs sous-documents.
- NAV : représente une section de liens vers d'autres pages ou des fragments de cette page
- ASIDE : représente une section de la page dont le contenu est indirectement lié à ce qui l'entoure et qui pourrait être séparé de cet environnement
- HEADER : représente un groupe d'introduction ou une aide à la navigation. Il forme l'entête d'un document. Il peut contenir des éléments de titre, mais aussi d'autres éléments tels qu'un logo, un formulaire de recherche, etc.
- FOOTER : représente le pied de page, ou de la section, ou de la racine de sectionnement la plus proche

La figure 3.4 montre un découpage explicite de la structure avec HTML 5 en opposition au découpage implicite de HTML 4 montré dans la figure 3.2.

3.1.3 ARIA

ARIA (Acessible Rich Internet Application) [7] est la spécification d'une Ontologie décrivant une interface graphique. Elle fournit des informations sur la structura-

tion d'un document et plus généralement elle décrit les éléments qui composent une interface graphique au moyen d'un ensemble de rôles, d'états et de propriétés.

Rôle Les rôles permettent d'identifier la fonction de chaque élément d'une interface. Ils sont regroupés en trois catégories :

- Widget Roles : définit un ensemble de widgets (alertdialog, button, slider, scrollbar, menu,etc.)
- Document Structure Roles : décrit les structures qui organisent un document (article, definition, entête,etc.)
- Landmark Roles : décrit les régions principales d'une interface graphique (main, navigation, search,etc.)

États et propriétés ARIA prend en compte l'aspect dynamique et interactif des éléments d'une interface. Elle permet d'associer des états et des propriétés aux éléments d'une interface. Un état est une configuration unique d'un objet. Par exemple, on peut définir l'état d'un bouton par l'état *aria-checked* qui peut prendre trois propriétés suivant l'interaction avec l'utilisateur : *true* - *false* - *mixed* . Dans le cas d'une checkbox, *true* indique si la checkbox est cochée, *false* si elle ne l'est pas et *mixed* dans le cas d'un ensemble de checkbox indique que certaines sont cochées.

Aria prévoit même un système d'annotation pour les objets ayant des comportements asynchrones. Par exemple, on peut indiquer par une annotation qu'un élément se met à jour de manière autonome.

3.1.4 CSS

CSS est un langage de feuille de style qui permet aux auteurs des pages web de lier du style aux éléments HTML. Le style définit comment afficher un élément (ex. les polices de caractères, l'espacement, les couleurs, *etc.*). CSS permet ainsi de séparer la présentation du style du contenu (*cf.* figure 3.5). L'avantage est une simplification de l'édition et de la maintenance d'une page web.

```
<style>
p.serif{font-family:"Times New Roman",Times,serif;}
p.sansserif{font-family:Arial,Helvetica,sans-serif;}
</style>

<body>
<p class="serif">Ceci est un paragraphe
avec un style de font Times New Roman font.</p>
<p class="sansserif">Ceci est un paragraphe
avec un style de font the Arial font.</p>
</body>
</html>
```

Ceci est un paragraphe avec un style de font Times New Roman font.

Ceci est un paragraphe avec un style de font the Arial font.

FIGURE 3.5 – Exemple CSS

Modèle de boîte CSS génère pour chaque élément de l'arbre du document (DOM) une boîte rectangulaire. Les boîtes rectangulaire sont conformes à un modèle de boîte et sont agencées suivant un modèle de mise en forme décrit en section 3.1.4

Chaque boîte possède ainsi une aire de contenu (*e.g* une texte, une image, *etc.*) entourée en option par une aire d'espacement, une aire de bordure et une aire de marge (*e.g* figure 3.6).

Modèle de mise en forme Chaque boîte se voit attribuer un type qui affecte en partie son positionnement. Les deux principaux types sont les *boîtes en bloc* et *les boîte en-ligne*. Les éléments de type bloc sont des éléments dont le rendu visuel forme un bloc (*e.g* figure 3.7 avec l'élément de paragraphe <p>). Les éléments de type en-ligne sont des éléments qui n'ont pas la forme de blocs de contenu (*e.g* figure 3.7 avec l'élément). Les boîtes en-ligne sont placées horizontalement, l'une après l'autre, en commençant en haut du bloc conteneur. Les blocs conteneurs sont des boîtes qui encapsulent d'autres boîtes. Les boîtes en-bloc sont placées l'un après l'autre, verticalement, en commençant en haut du bloc conteneur. Le schéma de positionnement décrit est appelé *flux normal*.

Une fois le *flux normal* calculé, il est possible de le modifier.

Un premier mécanisme possible est le positionnement relatif. La position de la boîte est exprimée en propriété de décalage par rapport à son bloc conteneur :

– 'top' : définit le décalage du bord haut de la marge d'une boîte sous le bord

haut de la boîte du bloc conteneur.
- 'right' : définit le décalage du bord droit de la marge d'une boîte à gauche du bord droit de la boîte du bloc conteneur.
- 'bottom' : définit le décalage du bord bas de la marge d'une boîte au-dessus du bord bas de la boîte du bloc conteneur.
- 'left' : définit le décalage du bord gauche de la marge d'une boîte à droite du bord gauche de la boîte du bloc conteneur.

Un deuxieme mécanisme est le positionnement flottant. Une boîte flottante est déplacée vers la gauche ou la droite sur la ligne courante du *flux normal*. Le contenu du document s'écoule alors le long des flancs de cette dernière.

Un troisième mécanisme est le positionnement absolu. La boîte est retirée du *flux normal* et est positionnée par rapport à son bloc conteneur. La différence avec le positionnement relatif est que le positionnement de la boîte n'a aucun effet sur les boîtes du même niveau de parenté. Ces boîtes peuvent, ou non, cacher les autres boîtes.

Avant-plan et d'arrière-plan Les propriétés CSS permettent aux auteurs la spécification d'une couleur d'avant-plan et d'arrière-plan pour un élément. La couleur d'arrière-plan peut être une couleur ou une image. L'arrière-plan correspond aux aires de contenu, d'espacement et de bordure. La couleur d'avant-plan correspond à la couleur du contenu de texte d'un élément.

Les polices CSS permet de spécifier l'utilisation de plusieurs représentations pour les caractéristiques des textes : *la police*. Une liste exhaustive de propriétés permet de spécifier la police d'un élément contenu dans une boîte. On peut spécifier par exemple une famille de polices (serif, sans-serif), le style de la police (italic, oblique), la taille, *ect.*

Les textes CSS définit la représentation visuelle des caractères, des caractères blancs, des mots et des paragraphes. On peut spécifier un alinéa pour la première ligne du texte dans un bloc ('text-indent'), l'alignement d'un contenu en-ligne dans un élément de type bloc ('text-align'), le comportement de l'espacement entre les caractères du texte ('letter-spacing'), *ect.*

3.2 Extraction de structures

Cette partie de l'état de l'art s'intéresse aux travaux permettant la découverte des différentes structures logiques qui composent une page web. Nous présentons les approches puis nous en présentons une synthèse.

3.2.1 Mapping

Une opération de *mapping* est une opération qui identifie une correspondance entre deux structures, ici des arbres. Une page HTML peut-être vue comme un arbre dont les sommets sont étiquetés et ordonnés. En effectuant un *mapping* entre deux DOM de pages web il est possible de découvrir des structures logiques. Nous donnons ci-dessous une définition formelle d'un *mapping* (introduite par [12]).

Definition. Soit T_x un arbre et soit $t_x[i]$ le i-ième sommet de T_x dans un parcours préfixe. Un *mapping* M de l'arbre T_1 de taille n_1 vers l'arbre T_2 de taille n_2 est un ensemble de paires ordonnées d'entiers (i, j), $1 \leqslant i \leqslant n1$, $1 \leq j \leq n2$, satisfaisant les conditions suivantes, pour tous $(i1, j1), (i2, j2) \in M$:

- $i_1 = i_2$ si et seulement si $j_1 = j_2$ (one-to-one condition)
- $t_1[i_1]$ est à gauche de $t_1[i_2]$, si est seulement si $t_2[j_1]$ est à gauche de $t_2[j_2]$ (préservation de l'ordre des nœuds frères)
- $t_1[i_1]$ est un ancêtre de $t_1[i_2]$ si et seulement si $t_2[j_1]$ est un ancêtre de $t_2[j_2]$ (préservation de l'ordre des ancêtres)

La première condition établit qu'une paire de sommet ne peut apparaître plus d'une fois dans le *mapping*. La seconde condition impose une préservation de l'ordre des nœuds frères et la dernière impose une relation d'héritage entre les nœuds de l'arbre. La figure 3.8 illustre un *mapping* entre deux arbres. De manière intuitive, un *mapping* est la description d'une suite d'opérations d'édition (substitution, suppression et insertion) qui permet la transformation d'un arbre vers un autre. Les opérations d'édition se définissent ainsi :

Notons Λ un nœud vide. Une opération d'édition est écrite $b \rightarrow c$, où b et c sont soit un nœud, soit Λ. Alors :

- $b \rightarrow c$ est une opération de substitution si $b \neq \Lambda$ et $c \neq \Lambda$
- une opération de suppression si $b \neq \Lambda \doteq c$
- une opération d'insertion si $b = \Lambda \neq c$

En appliquant différentes restrictions sur les opérations d'édition dans un *mapping*, plusieurs classes de *mapping* peuvent se distinguer modélisant différentes applications pratiques. Les auteurs de [13] définissent une classe de *mapping* particulière :

un *mapping* descendant restrictif. Ce derniers restreint les opérations d'édition aux les feuilles de l'arbre. Ce *mapping* permet d'identifier les sous-structures communes entre deux arbres.

Definition. Un *Mapping* M de l'arbre T_1 vers l'arbre T_2 est descendant si pour toute paire $(i_1, i_2) \in M$ il y a aussi une paire $(parent(i_1), parent(i_2)) \in M$

Definition. Un *Mapping* descendant M de l'arbre T1 vers l'arbre T2 est restrictif si et seulement si $(i_1, i_2) \in M$ telle que $t_1[i_1] \neq t_2[i_2]$ il n'y a pas de descendant de i_1 ou i_2 dans M

Dans leur approche [13] s'intéresse à l'élimination de certaines structures logiques (publicités, bannières, etc.) dégradant les résultats des algorithmes des moteurs de recherche. Le postulat étant que les scripts de publication de contenu automatique génèrent, pour les structures nuisibles (bannière, publicité,*etc.*), les mêmes constructions d'une page à l'autre. Les auteurs s'intéressent à l'identification des structures logiques dans les pages web au moyen de construction de template matchant ces structures. La construction des templates repose sur un processus de *mapping* descendant restrictif à partir d'un échantillon de pages issu d'une collection de page. L'algorithme proposé par les auteurs construit des templates correspondant aux sous-structures communes identifiées par *mapping* descendant restrictif entre les pages web. Soit une constante $MaxPage$, le nombre de pages pour construire un template (en moyenne une douzaine) et n la taille d'un arbre. La construction d'un template coûte $O(n^2)$. L'opération de recherche d'un template dans une page coûte $O(n)$.

3.2.2 Segmentation

Dans cette partie, nous introduisons plusieurs approches de segmentation. Dans le contexte des pages web, une opération de segmentation consiste à rassembler les nœuds du DOM suivant des propriétés prédéfinies. Le but étant de proposer une structure pour la page web traitée.

Segmentation par pattern de page

Les auteurs de [11] proposent dans leur papier une segmentation suivant un pattern de présentation défini a priori. Cette approche s'intéresse aux coordonnées cartésiennes des structures logiques dans une page web. L'idée étant que les concepteurs de page web suivent un même pattern de présentation, ce qui permet de faire un regroupement des structures logiques suivant leur localisation.

Sur la base d'observation, les auteurs proposent un pattern de présentation de page web (*cf. figure* 3.9). Le modèle construit est le suivant :

	Bannière	Pied de page	Menu latéral gauche	Menu latéral droit
Mauvais	41%	30%	21%	19%
Bon	10%	15%	3%	2%
Excellent	49%	55%	76%	23%

TABLE 3.1 – Tableau évaluation méthode segmentation par pattern

- une entête correspondant à l'emplacement de la bannière d'une page (H),
- un pied de page (F)
- une marge latérale gauche contenant des menus (LF)
- une marge latérale droite contenant des menus (LR)
- le centre de la page encapsulant le contenu principal de la page ((C))

La figure 3.9 modélise le partitionnement défini par les auteurs avec *W1* et *W2* définissant respectivement la marge contenant les menus latéraux gauches et la marge contenant les menus latéraux droits. Chacune d'elle représente 30% de la largeur de la page. *H1* et *H2* définissent respectivement l'entête de la page et le pied de page avec une hauteur de 200px pour H et 150px pour F.

Les coordonnées des structures sont calculées au moyen d'un moteur de rendu graphique.

En se basant sur ce modèle de représentation, des heuristiques sont déterminées afin de regrouper les nœuds du DOM de la page suivant le partitionnement de page proposé ci-dessus.

À partir de l'évaluation d'un corpus de 515 pages web, les résultats expérimentaux (publiés en 2002) montrent que le modèle proposé par les auteurs correspond bien à un modèle de présentation représentatif des pages web. Une note de phBon est attribuée aux zones dont les objets HTML sont à plus de 50% correctement étiquetés, c'est-à-dire que par exemple les éléments classifiés comme bannière correspondent bien à une bannière. Une note de *Excellent* est attribuée aux zones dont les objets HTML sont à plus de 90% correctement étiquetés. Une note de *Mauvais* est attribuée lorsque les critères précédents ne sont pas satisfaits

Segmentation par densitométrie textuelle

Les auteurs [10] cherchent à distinguer les différentes informations qui structurent la page en se basant sur l'analyse de la répartition de la densité textuelle dans une page. L'hypothèse des auteurs est que la densité des différents blocs de textes dans le DOM est une propriété suffisante pour identifier la structure logique d'une page.

La première étape consiste à identifier les différents segments textuels atomiques de la page. La page est vue comme une séquence d'éléments de caractères entrelacés de tags HTML. Certains tags HTML segmentent le flux textuel et d'autres non. Cette décision est prise non pas d'après la sémantique des balises mais d'après le rythme des séquences de caractères dans le flux. Par exemple si le flux passe d'une séquence de caractères courte à une séquence de caractères longue cela signale un nouveau segment. Si on analyse une séquence de textes courtes comme *<a>Accueille<a>New<a>Contact* suivie d'une séquence de texte longue comme * Une sequence de texte beaucoup plus longue * va donner naissance à deux nouveaux segments. Pour chaque segment (bloc) identifié par le processus ci-dessous, les auteurs vont calculer une densité textuelle $p(b_x)$ avec b_x un segment. Cette propriété correspond au ratio ci-dessous :

$$p(b_x) = \frac{NombreDeMotsDansb_x}{NombreDeLignesDansb_x}$$

Le nombre de lignes correspond aux nombres de séquences de 80 caractères possibles dans le segment traité. 80 étant une valeur fixée par les auteurs, elle correspond à la taille d'une phrase moyenne (pour les pays anglophones).

La seconde étape permet de construire des blocs dans la page correspondant à la structuration logique de la page à partir des segments identifiés dans l'étape précédente. Cette construction consiste à fusionner les blocs contingents en comparant leurs densités respectives. La fonction est définie ci-dessous :

$$\Delta p(b_x, b_y) = \frac{|p(b_x) - p(b_y)|}{max(p(b_x), p(b_y))}$$

Ce processus de grossissement est répété tant que la fonction définie ci-dessus ne satisfait pas un certain seuil. Cela conduit à la construction de blocs de forts granularité correspondant à la structuration logique de la page. La figure 3.10 présente un exemple de structure découverte d'une page avec l'approche décrite.

Segmentation par indice visuel

L'approche proposée par les auteurs [3] présente un algorithme de partitionnement basé sur les éléments de mise en forme des pages web. Le partitionnement extrait une structure qui regroupe les éléments d'une page d'après leur mise en forme (*e.g.* figure 3.11). Le postulat est que les éléments d'une page possédant des caractéristiques de mise en forme proches, telles que la police, la couleur, la taille, peuvent être regroupés.

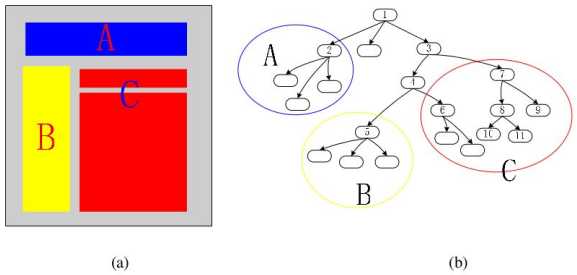


FIGURE 3.11 – Exemple de partitionnement, (a) page (b) DOM de la page [3]

L'algorithme exploite le Document Object Model [1] de la page web. Le processus de segmentation, figure 3.12, se décompose en trois phases : un processus d'extraction de blocs, un processus de détection de séparateur et un processus de reconstruction.

Le processus d'extraction détecte les éléments du niveau courant du DOM susceptibles de former un contenu cohérent. Cette détection repose sur des séparateurs explicites : on sait que certains éléments délimitent le contour d'un contenu (par exemple les balises <DIV>). Mais elle repose également sur une fonction de distance visuelle comparant les nœuds parents et frères du nœud courant : une balise <DIV> a de grandes chances de délimiter un contenu sémantiquement différent du nœud parent si la couleur de fond est différente de celle de ce dernier. Pour chaque nœud, l'algorithme vérifie s'il forme un bloc ou non. Si oui, il associe un degré de cohérence au bloc. Ce degré de cohérence est un indicateur de l'importance sémantique du bloc. Si non, il est appliqué le même processus aux enfants du nœud. Quand tous les nœuds du bloc courant sont extraits, ils sont mis dans un pool.

Des séparateurs entre les blocs sont ensuite détectés. L'algorithme détecte ici des séparateurs implicites, c'est-à-dire n'apparaissant pas dans la structure HTML. L'opération du calcul de séparateurs a une complexité de $O(n^2)$. Les séparateurs implicites sont les espaces entre les blocs d'un pool. Un poids est attribué à chaque séparateur suivant son importance (par exemple, plus l'espacement entre deux blocs est grand, plus le poids sera élevé). Ce poids est un indicateur de différence sémantique entre les blocs adjacents. Plus le poids du séparateur est élevé entre deux blocs, plus leur contenu sera sémantiquement éloigné.

Une construction hiérarchique des blocs est créée. Cette construction hiérarchique repose sur le degré de cohérence attribué à chaque bloc.

Pour chaque nouveau bloc de la structure hiérarchique construite, l'algorithme teste le degré de cohérence attribué par rapport à un seuil de cohérence défini. Ce seuil

1. Document Object Model

est défini suivant la granularité de la structure que l'on veut en sortie de l'algorithme. Si le degré de cohérence n'est pas supérieur au seuil de cohérence, le bloc est de nouveau proportionné. La structure finale est construite après que tous les blocs soient traités.

3.2.3 Synthèse

L'approche par correspondance de structure (section 3.2.1) entre les pages permet d'isoler seulement des fragments de page, la structure globale de celle-ci n'apparaît pas. Les approches par segmentation sont plus intéressantes car elles proposent un découpage reflétant une structure globale. La segmentation par pattern (section 3.2.2) est une approche efficace dans la plupart des cas. Cependant toutes les pages web ne suivent pas le même modèle de présentation. La segmentation par densitométrie (section 3.2.2) est une méthode indépendante du pattern de présentation de la page. Cependant, le problème avec cette approche est qu'elle ne fournit pas le moyen de prendre en compte les possibles écarts entre la structure décrite par HTML et le rendu final. C'est-à-dire que des nœuds du DOM peuvent être proches dans la structure du DOM puis avec l'application de propriétés de mise en forme, très éloignés avec le rendu visuel en particulier avec les propriétés de positionnements absolues décrites en section 3.1.4. La segmentation par indices visuels (section 3.2.2) comble les faiblesses des deux méthodes précédentes. L'inconvénient de cette approche se situe au niveau de l'algorithme de calcul des séparateurs qui est une opération coûteuse $O(n2)$.

3.3 Annotation de structures

Les auteurs [4] proposent une approche afin de découvrir la fonctionnalité des éléments composant une page web. Les différents éléments sont des objets qui apportent une information ou réalisent une action. Ces objets peuvent soit être des objets basiques ou bien des objets composites. Les objets basiques et composites sont le résultat d'un prétraitement (non décrit dans le papier) qui retourne une structure arborescente dont les feuilles forment des objets basiques et les nœuds ayant des descendants forment des objets composites. Afin de qualifier chaque nœud, les auteurs proposent un ensemble de propriétés pour attribuer une classe à chaque objet (basique et composite). L'hypothèse est que chaque objet au travers de ces fonctionnalités reflète l'intention de son auteur, c'est-à-dire sa sémantique.

La structure des objets basiques est décrie d'après les propriétés fonctionnelles ci-dessous :

Présentation : l'agencement (*e.g* alignement gauche), le type de média (*e.g* texte, image).

Sémantème : la sémantique associée par le HTML. Par exemple la balise *h1* définit un titre

Décoration : une valeur entre 0 et 1 qui définit dans quelle mesure l'objet sert à l'amélioration de l'esthétique de la page (lignes, séparateurs, et les objets avec des couleurs d'arrière plan).

Hyperlien : l'objet possède-t-il des hyperliens ? et vers où pointent-ils vers (domaine) ?

Interaction : Les types d'interactions de l'objet. Affichage, soumission d'informations, sélections etc.

Les propriétés listées précédemment sont calculées d'après l'analyse du code HTML. Par exemple l'attribut *href* (spécifie la destination d'un lien) est utilisé pour calculer la propriété d'hyperlien.

Les objets composites sont donc une composition d'objets basiques et/ou composites. Ils sont caractérisés par les propriétés décrites plus haut et les relations des objets qui le composent. Ces relations sont classées ainsi :

Relation de Classification :

- Complémentaire : les enfants de la racine de l'objet composite sont complémentaires à la réalisation de la fonction. Ils ne possèdent pas les mêmes propriétés basiques
- Parallèle : les enfants de la racine de l'objet composite ont une importance égale à la réalisation du but. Ils possèdent les mêmes propriétés basiques
- Présentation : verticale ou horizontale (*e.g* une liste à puces peut-être présentée horizontalement ou verticalement)

Une fois les propriétés ci-dessus établies, les auteurs définissent un ensemble de classes pour les objets :

Informatif : objets qui présentent le contenu informatif

Navigation : objets qui déclenchent une navigation

Interaction : objets qui fournissent des interactions avec l'utilisateur

Décoration : objets qui remplissent un rôle esthétique (puce de liste, séparateur, *etc.*)

Les classes des objets basiques sont déterminées simplement par la valeur de ces propriétés. Un objet basique informatif possède un contenu textuel important, une image (taille significative par rapport à l'objet). La classe des objets composites

intègre les propriétés de relation. Par exemple, on peut définir un menu (appartenant à la classe des objets de navigation.) comme suit : un objet composite dont les enfants de la racine sont des objets de navigation parallèle (même propriété), le nombre d'enfants navigables est supérieur à un certain seuil $Hmin$ et la propriété de texte est inférieure à un seuil $Lmax$

```
<body>
        <header></header>
        <nav></nav>
        <section>
                <article></article>
                <article></article>
        </section>
        <aside></aside>
        <footer></footer>
</body>
```

FIGURE 3.4 – Exemple d'attribution de rôle

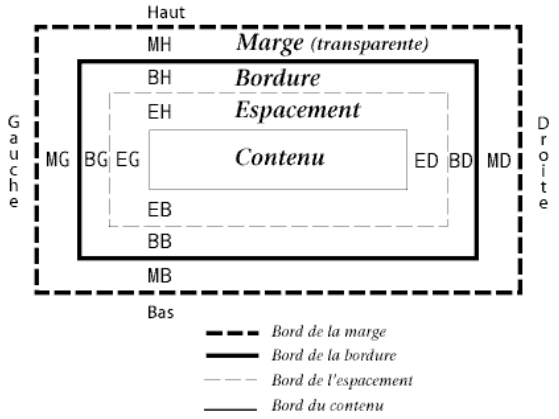


FIGURE 3.6 – Modèle de boîte

```
<p>Avant de faire le truc X il est
<strong>nécessaire</strong> de faire le truc Y avant.
</p>
```

FIGURE 3.7 – Exemple d'élément en-line

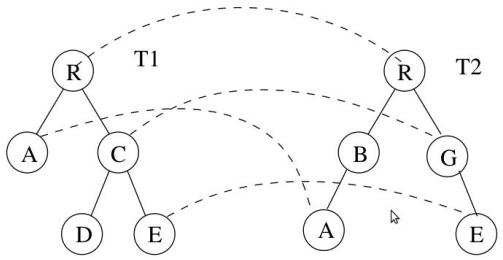


FIGURE 3.8 – Mapping entre deux arbres étiquetés et ordonnés

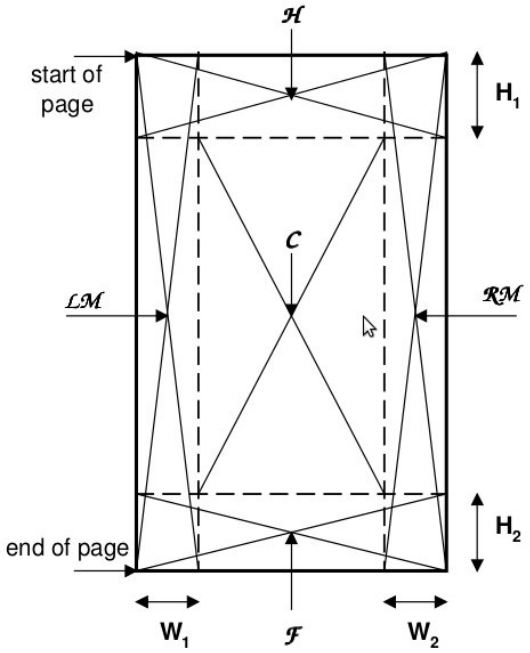


FIGURE 3.9 – Pattern de mise en forme de page web

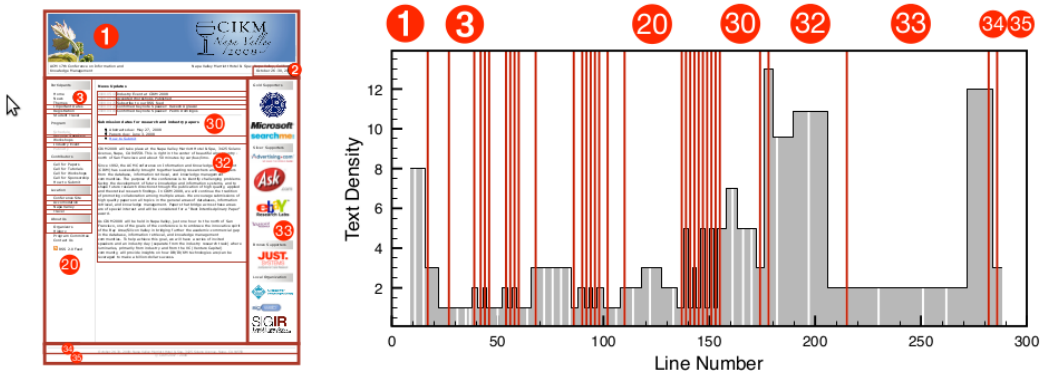


FIGURE 3.10 – Segmentation densitométrique [10]

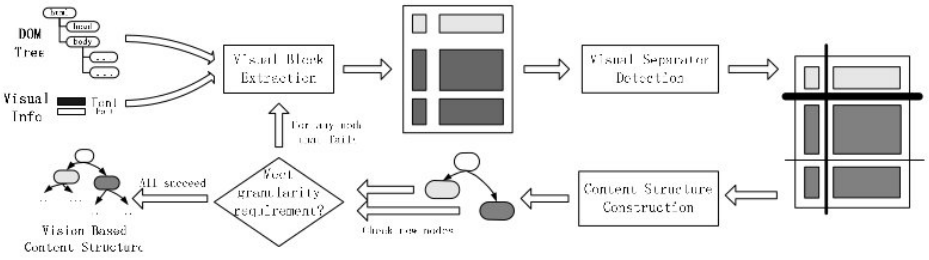


FIGURE 3.12 – Algorithme de segmentation [3]

Chapitre 4

Réalisations

Ce chapitre présente les contributions de ce stage. La première contribution est la réalisation d'un méta-modèle de page web utile au processus d'adaptation de page web décrit en chapitre 2. Puis nous décrivons la conception d'une méthode d'extraction des structures d'une page web et enfin des propositions pour annoter les structures extraites par des annotations conformes aux éléments de notre méta-modèle.

4.1 Méta-modèle

HTML 4 fournit une sémantique pour définir des nœuds de contenu (image, texte) mais pas pour des structures logiques plus élaborées (article, menu, bannière etc.). Par exemple, la figure 4.6 présente deux structures logiques extraites d'une page du site web *lemonde.fr* et d'une page du site web *www.eclipse.org*. D'après cette figure, on sait que les éléments textuels sont des liens de navigation signalés par la balise *<a href=...>*, mais il n'y a pas d'informations sémantiques sur la structure qu'ils composent (un menu de navigation). Ce premier point pose un problème pour l'expression de préférences sur de telles constructions.

La contribution de ce méta-modèle est d'apporter une composante (dimension) sémantique aux langages de publication du web. Cette sémantique est la concrétisation des différents concepts définis par les auteurs de pages web et ce qu'en comprend le lecteur. Dans un premier temps cette composante est essentielle pour l'expression des souhaits de personnalisation de l'utilisateur. En second lieu, le méta-modèle nous fournit une couche d'abstraction permettant de s'affranchir de la diversité de représentation des données.

Les parties suivantes proposent notre modélisation de HTML 4 et 5 puis introduisent le méta-modèle que nous avons conçu.

4.1.1 Méta-modèle HTML 4 et 5

Les deux méta-modèles présentés ci-dessous sont réalisés d'après notre synthèse faite de HTML 4 et 5 dans le chapitre 3.1.

HTML 4

Méta-modèle HTML 4 HTML 4 discrimine ses éléments de langage suivant que pendant l'affichage du flux de contenu les éléments produisent un retour à la ligne ou non (*cf.* section 3.1.4). Le modèle de HTML 4 repose plutôt sur une description des aspects de mise en forme des éléments que sur la sémantique des structures construites. Nous proposons dans la figure 4.1 un méta-modèle de HTML 4. Les principales méta-classes sont :

- La méta-classe *Bloc* dont les instances provoquent un retour à la ligne. Par exemple, une division ( ) ou un paragraphe ( ) génère un retour à la ligne
- La méta-classe *Inline* dont les instances ne provoquent pas de retour à la ligne. Par exemple, * Bonjour tous le monde *, la balise ** ne provoque pas de retour à la ligne dans le paragraphe.

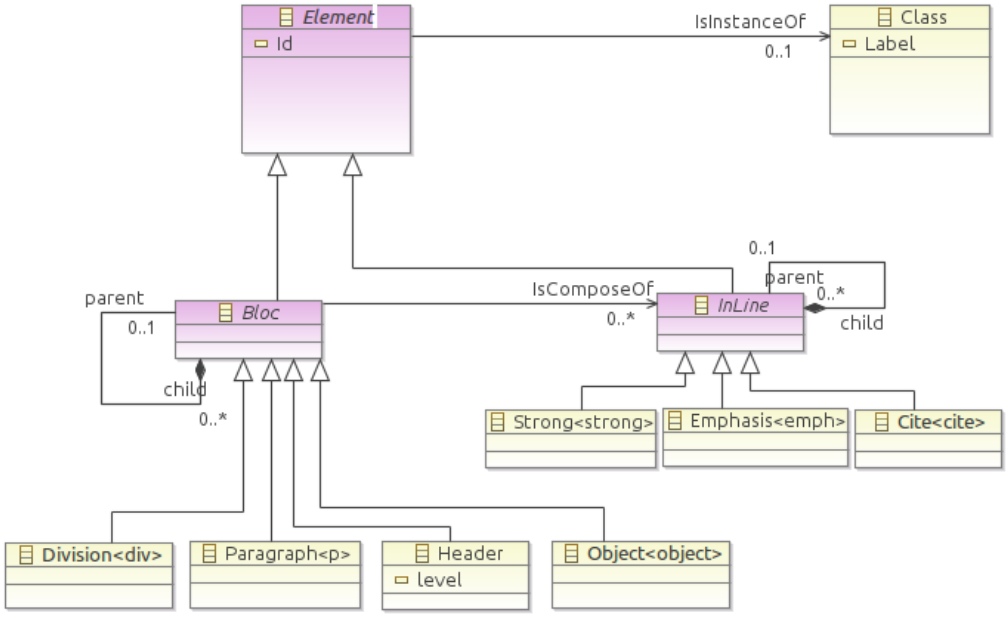


FIGURE 4.1 – Méta-modèle HTML 4

Expression de préférences HTML 4 délimitent les structures logiques de plus haut niveau à l'aide d'éléments structurels génériques (*div*) et leurs propriétés de mise en forme (bloc/inline). Si l'on instancie le premier menu de la figure 4.6, on voit bien que HTML 4 ne possède pas un niveau d'expressivité suffisant pour exprimer une préférence, par exemple, sur un menu de navigation.

HTML 5

Méta-modèle HTML 5 HTML 5 propose une classification plus élaborée qui ne se concentre plus sur les aspects de mise en forme mais sur la sémantique des différentes structures. Nous proposons dans la figure 4.2 un méta-modèle de HTML 5. Ci-dessous les principales classes de balises :

- Les instances de la méta-classe **Sectioning** définissent le contenu comme des éléments qui créent une nouvelle section dans le plan d'un document. Les instances de <section>, <article>, <nav>, <aside> héritent de cette dernière.
- Les instances de la méta-classe **Phrasing** définissent les éléments de langage qui peuvent définir un texte. Par exemple, la méta-classe *Strong* définit un texte important.

- Les instances de la méta-classe **Embedded** définissent un contenu importé d'un autre espace de nom. C'est le cas par exemple des instances de la méta-classe *Video*, son contenu est importé de l'extérieur.
- Les instances de la méta-classe **Interactive** définissent un contenu conçu pour une interaction avec l'utilisateur. Par exemple, pour une instance de la méta-classe *Video*, l'utilisateur va pouvoir arrêter ou mettre en pause la vidéo.

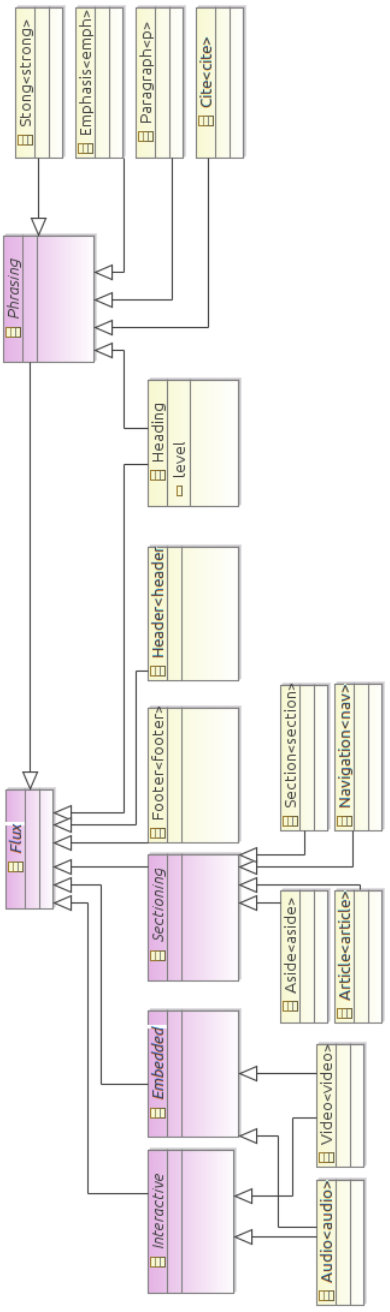


FIGURE 4.2 – Méta-modèle HTML 5

Expression de préférences Si l'on compare le méta-modèle de HTML 4 avec celui de HTML 5, on constate dans un premier temps que la mise en forme ne fait plus partie du méta-modèle. Le concept d'éléments en bloc et en-ligne est délégué à CSS. Dans un second temps il intègre des éléments de langage spécifique (et non plus générique) pour la description des structures construites. Si l'on instancie le premier menu de la figure 4.6, avec des éléments de HTML 5, il est possible d'exprimer, par exemple, une préférence sur l'accessibilité des menus de navigation dans une page.

Une préférence sur l'accessibilité serait, par exemple, que la taille du texte du menu de navigation soit supérieure à 15 pixels :

```
context Navigation inv:
        self.style.size < 15px
```

4.1.2 Méta-modèle conçu pour le projet

Modèle de contenu

La conception de cette partie du méta-modèle repose sur les éléments conceptuels identifiés dans la sous-section 4.1.1 en étendant l'expressivité de notre méta-modèle aux *Landmarks* de ARIA (Bannière, Contenu principal, *etc.*) et aux éléments de langage décrivant les éléments d'interaction.

Chaque élément du méta-modèle (*cf.* figure 4.3 et 4.4) construit va nous permettre de qualifier les éléments d'une page. Une première partie décrit les éléments de structuration.

Les principales méta-classes de structures sont :

- Les instances de *REGION* correspondent à un regroupement par thématique des éléments qu'elles encapsulent, c'est-à-dire des éléments que l'on peut regrouper sur la base d'une information commune.
- Les instances de la classe *LANDMARK* spécifient les instances de la classe *REGION* en proposant des thématiques récurrentes dans la conception des pages web actuelles. On a par exemple la méta-classe *NAVIGATION* pour les menus ou la méta-classe *BANNER* pour la bannière d'une page web.
- Les instances de *SECTIONHEAD* synthétisent le contenu d'instances de *SECTION*. Elles possèdent une portée locale, celles-ci synthétises le contenu de l'instance de type *SECTION* le plus proche.
- Les instances de *Texte* définissent un contenu textuel (*e.g* une emphase, une citation, *etc.*). Elles ne peuvent contenir que des éléments textuels ou des instances de la méta-classe *Texte*.

Une seconde partie de ce méta-modèle décrit les éléments d'interaction dans une page web. Les principales méta-classes sont :

- Les instances de *INPUT* qui permettent des entrées utilisateurs. Elles correspondent aux balises de type *<input>* de HTML4.
- Les instances de *COMMAND* qui réalisent des actions mais ne reçoivent pas d'information en entrée. Par exemple les balises de type *<a href=...>* déclenchent une action de navigation ou une balise de type *<input type=button>* déclenche l'envoi de données d'un formulaire vers un serveur.
- Les instances de *SELECT* qui permettent de faire des sélections parmi un ensemble de choix.

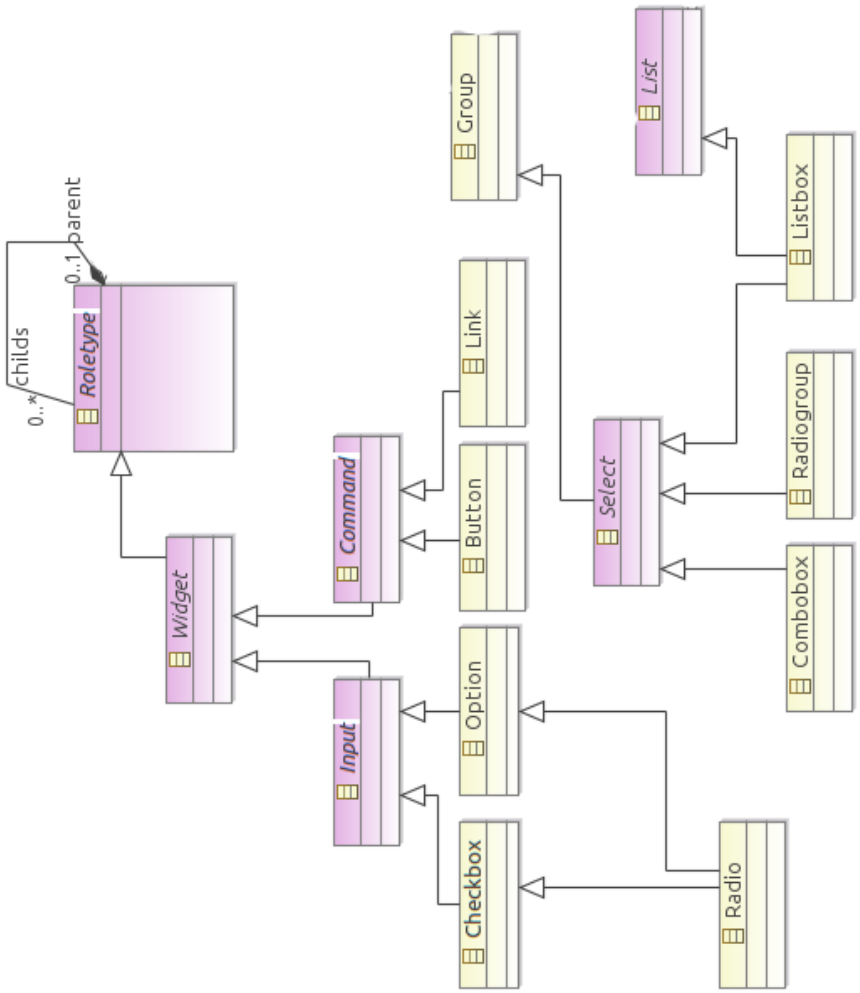


FIGURE 4.3 – Méta-modèle interaction

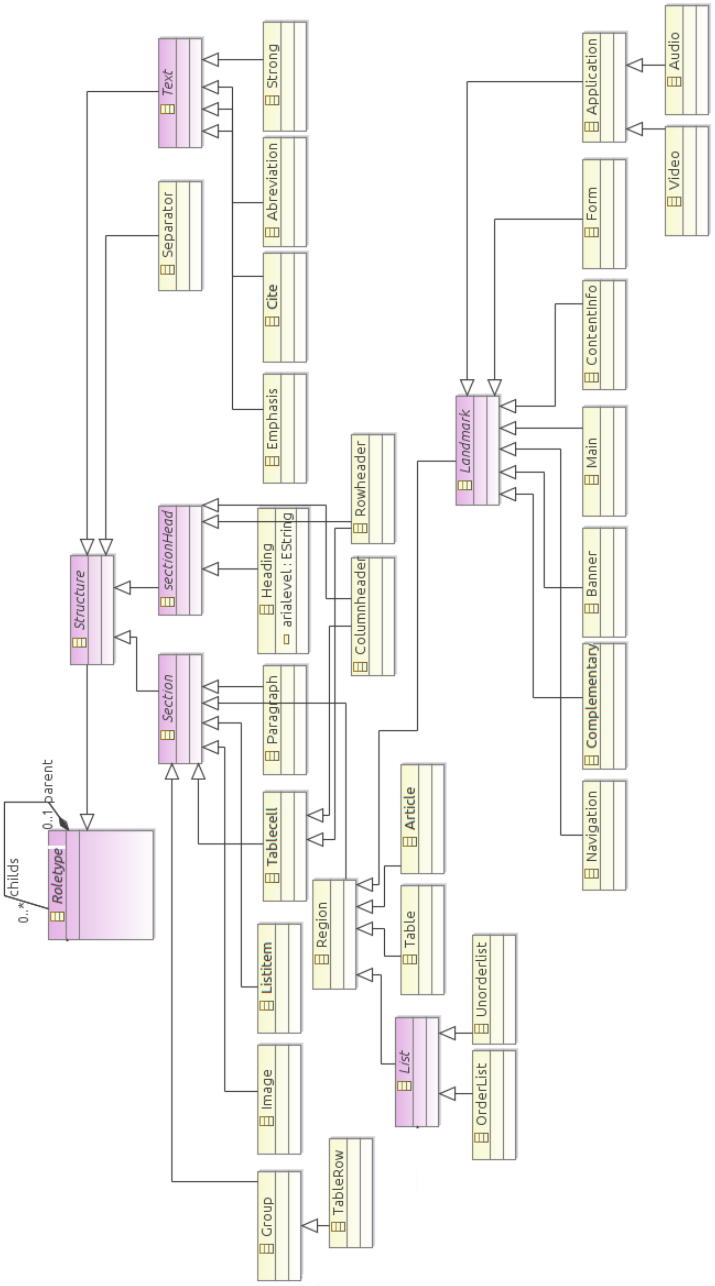


FIGURE 4.4 – Méta-modèle contenu

Modèle de mise en forme

Le méta-modèle de mise en forme (*cf.* figure 4.5) repose sur les éléments du langage CSS. Cette partie est un support essentiel à l'expression des préférences, en particulier pour les préférences liées à l'accessibilité. Les principales méta-classes sont :

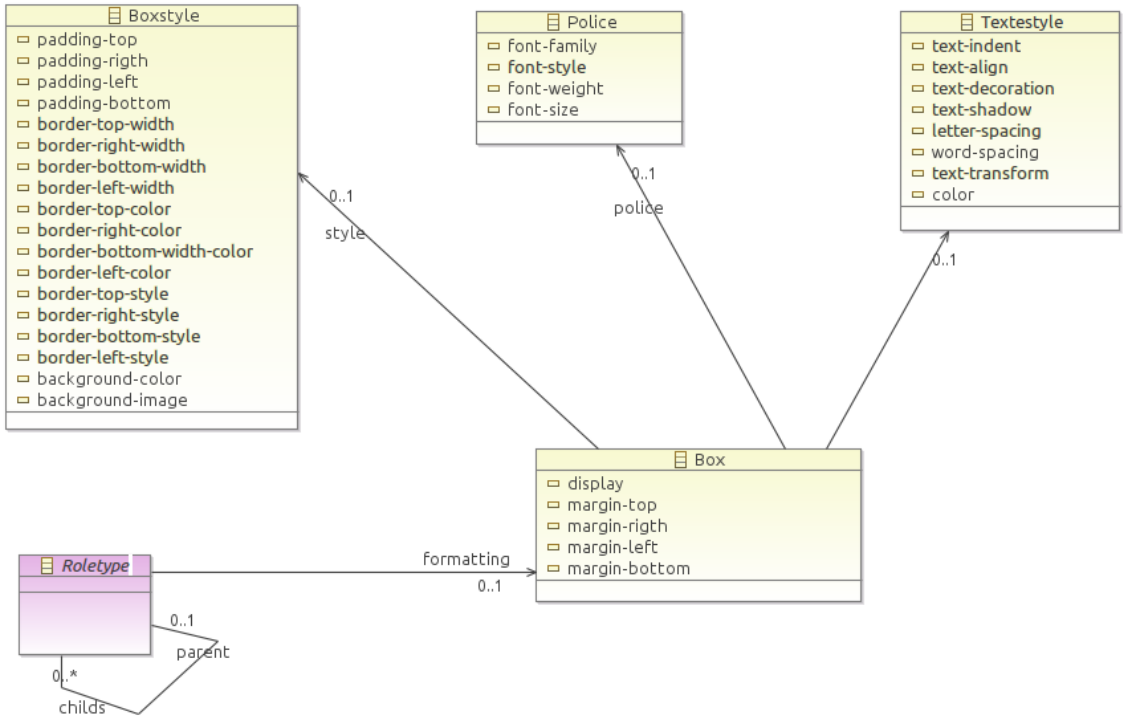


FIGURE 4.5 – Méta-modèle

- Les méta-classes *BOX* qui représentent le concept de bloc conteneur. Elles vont permettre d'exprimer des préférences sur l'accessibilité telles que l'espacement plus ou moins fort (propriété padding) avec les éléments connexes à ces dernières. Mais encore sur les contrastes avec la propriété de couleur d'arrière plan et la propriété de couleur de la méta-classe *Text*.
- Les méta-classes *TEXT* décrivent la représentation visuelle des caractères, mots et paragraphes contenus dans une boîte. Par exemple, la propriété *letter-spacing* spécifie l'espace entre les lettres.

– Les méta-classes *POLICE* décrivent la représentation visuelle des caractères. Dans le contexte de l'amélioration de l'accessibilité il peut être intéressant, par exemple, d'éviter certains types de *Font* qui peuvent rendre un contenu plus illisible pour certaines pathologies.

Expression de préférences

On voit d'après les préférences exprimées ci-dessous que le méta-modèle permet de formuler des préférences plus contextualisées. Le premier exemple exprime que le texte rouge d'un paragraphe du contenu principale ne peut pas avoir de fond vert. Le second exemple exprime que l'on veut masquer la bannière.

```
context Main inv:
        if Paragraph.style.color = red implies
                Paragraph.style.background-color != green
```

Exprimer des préférences sur des structures plus haut niveau

```
context Banner inv:
        self.style.display = hidden
```

4.1.3 Discussion

La réalisation de ce méta-modèle répond aux premiers besoins identifiés pour l'expression de souhait de personnalisation de page web pour en améliorer l'accessibilité. Cependant, d'autres aspects pourraient être modélisés comme une classification des pages web (*e.g* actualité, commerce en ligne, *etc.*) et l'écriture de transformations. L'utilisateur pourrait exprimer des préférences sur la présentation du site suivant son type. Par exemple, pour un site d'actualité, présenter un sommaire des titres des articles. Le méta-modèle ne traite que des aspects de présentation dans une page web. Des perspectives seraient la réification des aspects de navigation (fils de lecture dans le contenu) et les aspects dynamiques (interaction des éléments de la page avec l'utilisateur).

4.2 Extraction de structures

4.2.1 Introduction

Dans cette partie nous traitons l'extraction des structures logiques dans une page. Nous nous intéresserons à leur signification dans la section 4.3. Le DOM d'une page

web est constitué d'un ensemble de structures logiques, qui sont des regroupements d'éléments qui remplissent une fonction dans la page.

La première difficulté est que ces structures logiques ne sont pas annotées de manière explicite dans le DOM de la page. Nous voulons dire par là qu'il n'y pas d'annotations standards pour identifier et qualifier une structure. L'annotation se fait de manière implicite par l'attribution de classes d'appartenance (*cf.* section 3.1.1) permettant au navigateur web d'associer des mises en forme aux nœuds du DOM. La sémantique des classes n'est alors connue que par le concepteur.

Une seconde difficulté provient de l'hétérogénéité du contenu. Une page définit plusieurs contenus (bannière, pied de page, article, *etc.*). Si l'on regarde le listing 4.1, dans le contexte de notre méta-modèle, on doit identifier trois structures logiques différentes : une bannière, un document, un menu. Les éléments du langage HTML 4 ne permettent pas de signaler les différents contenus dans une page.

Listing 4.1 – Exemple de contenu hétérogène

```
<div id=page>
 <div id='banner'></div>
 <div id='menu-principal'></div>
 <div id='document'>
   <div>
         <h1>Introduction</h1>
         <p>une longue introduction (...)</p>
   </div>
   <div>
         <h1>Section 1</h1>
         <p>un long paragraphe (...)</p>
         <p>un second long paragraphe (...)</p>
   </div>
</div>
<div id='footer'></div>
</div>
```

Une troisième difficulté est la diversité des structures. HTML 4 possède une composante syntaxique extrêmement souple : HTML 4 autorise la construction d'un ensemble potentiellement infini de structures logiques. Si l'on reprend la figure 4.6, on voit que l'on a deux structures logiques différentes ayant la même sémantique (un menu). Cette diversité pose des difficultés pour l'extraction de structure puisqu'on ne peut pas appliquer une simple technique d'analyse syntaxique. Il est également possible de construire des structures non conformes aux spécifications du W3C ce qui écarte un découpage du DOM basé exclusivement sur les spécifications du W3C. Les concepteurs de page web ne tiennent pas toujours compte des spécifications des langages dans la création des pages. Par exemple, dans le listing 4.2 on constate que l'étiquette *P* n'est pas utilisée pour structurer un texte en paragraphe mais pour structurer des liens de navigation en menu.

Une quatrième difficulté est le possible écart dans la structure du DOM et le rendu final. Les nœuds dans le DOM peuvent être proches mais au moment de la mise en page par le navigateur peuvent être très éloignés. Ceci est dû aux possibilités offertes par CSS de modifier l'algorithme du moteur graphique des navigateurs (*c.f* 3.1.4).

Listing 4.2 – Extrait du menu lemonde.fr

```
<p>
        <a href='...'>Le Monde</a>
```

```
<div id="header-nav">
<ul>
        <li><a href='...'>Home</a></li>
        <li><a href='...'>Downloads</a></li>
        <li><a href='...'>Users</a></li>
        <li><a href='...'>Members</a></li>
        <li><a href='...'>Committers</a></li>
        <li><a href='...'>Resources</a></li>
        <li><a href='...'>Projects</a></li>
        <li><a href='...'>About Us</a></li>
</ul>
</div>
```

```
<div>
<p>
        <a href='...'>Le Monde</a><a>Télérama</a>
        <a href='...'>Le Monde diplomatique</a>
        <a href='...'>Le Huffington Post</a>
        <a href='...'>Courrier international</a>
        <a href='...'>La Vie</a>
        <a href='...'>au Jardin</a>
</p>
</div>
```

FIGURE 4.6 – Exemple de différentes conceptions de menu avec HTML4

```
        <a href='...'>Télérama</a>
        <a href='...'>Le Monde diplomatique</a>
        <a href='...'>Le Huffington Post</a>
        <a href='...'>Courrier international</a>
        <a href='...'>La Vie</a>
        <a href='...'>au Jardin</a>
</p>
```

Listing 4.3 – Pattern page d'accueil google.fr

```
<div class='style1'>
 <div class='style2'>
  <div class='style3'>
        <form>...</form>
  </div>
 </div>
```

```
</div>
```

Toutes ces contraintes posent le cadre de l'élaboration de notre méthode d'extraction des structures logiques dans une page. Plusieurs méthodes sont décrites dans l'état de l'art qui présente une synthèse des différentes approches (*cf.* 3.2.3). Nous choisissons de tester et d'améliorer une approche qui s'intéresse aux propriétés de mise en forme pour proposer un regroupement des nœuds du DOM reflétant des structures logiques. En effet, notre constat est que la structure d'une page est explicitée au moment de la mise en page (par le navigateur).

Les auteurs de l'approche [3] proposent un découpage du DOM en se basant sur l'élaboration d'heuristiques prenant en compte les propriétés de mise en forme. Cependant, la conception des heuristiques est fortement basée sur des pages web conçues suivant un pattern de conception reposant sur une organisation du contenu avec la balise TABLE. La balise TABLE est utilisée pour organiser l'interface de la page. Ce pattern de conception de page web est très peu utilisé actuellement. La raison principale est une grande ambiguïté dans l'interprétation du contenu (par une machine) puisque la sémantique de la balise TABLE a vocation de structurer une information et non l'organisation de la mise en page de la page. Nous supposons que la raison de cette évolution de la conception chez les développeurs est due au mauvais référencement par les moteurs de recherches qui peuvent entraîner ces ambiguïtés. Ainsi, l'application directe de ces heuristiques ne propose pas un découpage reflétant la structure logique d'une page dans la majorité des cas.

Notre méthode prend en compte de manière plus efficace la conception actuelle des pages web. Dans un premier temps, nous proposons plusieurs concepts pour l'élaboration d'heuristiques. Ces concepts définissent un modèle de page web intégrant les aspects de mise en forme et de structuration des éléments qui sont le support de la définition des heuristiques.

4.2.2 Méthode

Description du modèle de page

La méthode choisie est un processus itératif basé sur des découpages successifs de la structure de la page. Depuis la racine de la structure d'une page, on va parcourir (suivant un parcours préfixé) ses nœuds. Pour chaque nœud parcouru on va soit l'annoter comme une structure logique, soit le diviser. Ces décisions sont prises sur la base d'heuristiques définies a priori. Dans le premier cas on ré-applique le processus de découpage, dans le second cas on continue le parcours. Les itérations sont stoppées lorsque tous les nœuds atomiques sont identifiés. Le résultat est la construction d'un

arbre reflétant la composition des structures logiques dans la page.

Nous définissons ci-dessous les concepts utiles à l'élaboration des heuristiques. Nous proposons une classification des nœuds (étendue de [3]), une fonction de distance visuelle et une fonction de calcul du poids d'un nœud :

Nœuds conteneurs : l'ensemble des nœuds qui peuvent contenir d'autres nœuds (conteneurs et données). Ils ont un rôle de structuration logique. Par exemple, *<p>* peut structurer un ensemble de nœuds de données textuelles pour former un paragraphe. Ils correspondent aux nœuds avec la propriété display de CSS (display : 'bloc')

Nœuds de mise en forme : l'ensemble des nœuds qui n'ont pas un rôle de structuration mais viennent signaler une mise en forme aux nœuds de données. Par exemple les balises ** signalent de mettre en gras un texte mais ne structurent pas les nœuds qu'elles encapsulent. Si les nœuds conteneurs décrivent une structure logique, les nœuds de mise en forme décrivent une structure physique. Ils correspondent aux nœuds avec la propriété display de CSS (display : 'inline')

Nœuds de contenu : la classification de [3] propose des nœuds textes (feuilles du DOM qui contiennent du texte) mais pour l'élaboration de nos heuristiques nous avons fait évoluer cette classe de nœuds en nœuds de contenu. Ceux-ci regroupent les nœuds que nous qualifions de contenu atomique. Les contenus atomiques correspondent aux éléments multimédias dans la page : texte, image, video, audio, etc. On inclut les balises <img>, <object> et <h1-h6>.

Nœuds de contenus virtuels : l'ensemble des nœuds qui contiennent des nœuds de contenu encapsulés par des nœuds de mise en forme. Par exemple, *<P>Ceci est un paragraphe avec un nœud de mise en forme</P>*. *P* est un nœud virtuel

Nœuds de description : cette classe vient s'ajouter aux concepts décrits par [3]. Elle caractérise les nœuds qui donnent des méta-informations sur une page. Par exemple, la balise *<style>* renseigne sur la feuille de style à appliquer dans la page. Ce sont des balises utilisées pour renseigner le navigateur internet sur la page qu'il doit afficher.

Nœuds visibles : l'ensemble des nœuds qui sont visibles au travers du navigateur (en excluant les noeuds de description).

Fonction de distance visuelle : cette fonction compare les propriétés de mise en forme associées à chaque nœud pour déterminer s'ils sont visuellement distants ou non. Par exemple, la fonction renvoie vrai si les nœuds ne partagent pas les mêmes propriétés :

- d'arrière-plan,
- de bordure,
- etc.

Taille du nœud : les auteurs [3] définissent une taille pour chaque nœud suivant l'étiquette HTML qui lui est associée. Pour certaines balises, la grande souplesse syntaxique fait qu'une taille basée sur les étiquettes HTML n'est pas significative. C'est la raison pour laquelle nous calculons une taille qui prend en compte les éléments contenus par le nœud et leurs positions dans le DOM. Cette mesure est particulièrement utile pour déterminer si *div* est utilisée pour introduire une nouvelle structure ou non.
$TailleRelative(x) = \frac{TailleDunoeud_x}{100} * TailleDuNoeudRacine$

Heuristiques

L'élaboration des différentes heuristiques ci-dessous repose sur les concepts décrits précédemment.

règle 1 : si le nœud n'est pas un nœud de contenu et qu'il ne possède pas de nœuds visible alors on ne traite pas ce nœud.

La règle 1 filtre principalement les nœuds de description et éventuellement des balises oubliées par le concepteur (erreurs).

règle 2 : si le nœud possède un seul enfant visible et cet enfant n'est pas un nœud de données alors on parcourt ce nœud

La règle 2 permet de découvrir des structures logiques plus intéressantes dans une page. D'après l'exemple 4.4, cette règle va permettre d'atteindre le nœud *center* contenant la bannière de la page et la barre de recherche.

Listing 4.4 – Pattern page d'accueil google.fr

```
<span id='body'>
 <center>
   <div id='banner'>...</div>
   <div id='search-bar'>...</div>
 </center>
</span>
```

règle 3 : si les enfants du nœud sont des nœuds de données ou des nœuds de données virtuelles alors on annote ce nœud comme formant une structure logique.

La règle 3 nous permet de récupérer les nœuds de données. Ce sont les nœuds que l'on considère comme atomiques.

Listing 4.5 – Extrait de www.eclipse.org/actf/

```
<p>ACTF 1.1 including Visualization SDK is now available.
Please visit <a href="downloads/index.php">downloads page</a>
and get ACTF components!</p>
```

règle 4 : si l'un des enfants du nœud est un nœud conteneur et que sa taille est supérieure à un certain seuil alors on parcourt ce nœud

Si l'on reprend 4.4 après l'application de la règle 2 on va découvrir des structures logiques plus intéressantes qui sont la bannière de la page d'accueil de GOOGLE et la barre de recherche.

règle 5 : si l'un des enfants du nœud est étiqueté par *hr* alors on extrait les premiers nœuds visibles successeurs et prédécesseurs comme des structures logiques de chaque enfant étiqueté par *hr*.

La balise *hr* est employée pour marquer une séparation sémantique. Elle produit visuellement une ligne verticale dans le navigateur. Dans le listing 4.6, on va extraire deux structures (les deux sections encapsulées par les balises *div*).

Listing 4.6 – Exemple règle 5

```
<div>Ceci est une section sur une thématique 1.</div>
<hr>
<div>Ceci est une section sur une thématique 2</div>
```

règle 6 : si la somme des surfaces des enfants visibles du nœud est supérieure à ce dernier, alors on parcourt le nœud.

Dans le listing 5.1 le nœud *mw-navigation* est un nœud qui regroupe deux menus de navigation pour leur appliquer un style. Cette proximité des deux nœuds ne reflète pas le rendu final puisqu'après la mise en page les deux nœuds sont très éloignés (l'un dans l'entête de page, l'autre dans la marge gauche). On peut connaître cette information en regardant la surface des nœuds attribuée par le moteur graphique. En effet la surface du nœud identifiée par *mw-navigation* est inférieure à la somme de la surface des nœuds qu'il contient.

Listing 4.7 – Pattern wikipedia.fr

```
<div id="mw-navigation">
  <div id="mw-head">...</div>
  <div id="mw-panel">...</div>
</div>
```

règle 7 : si le premier enfant visible du nœud est un nœud de titre alors on extrait ce nœud comme une structure logique.

Les balises de titre sont utilisées pour introduire un nouveau contenu. Cela donne une indication sur l'introduction d'un nouveau contenu.

Listing 4.8 – pattern eclipse.org/actf

```
<div class="sideitem">
 <h6>Related links</h6>
  <ul>
  <li><a href="...">Project proposal</a></li>
  <li><a href="...">Creation Review Slides </a></li>
  <li><a href="...">1.0 Release &amp;amp; Graduation Review</a></li>
  </ul>
 </div>
```

règle 8 : si la fonction de distance visuelle est vérifiée (renvoie vrai) pour au moins un enfant du nœud alors on parcourt ce nœud et on extrait comme structure logique le nœud qui a vérifié la fonction de distance.

Dans le listing 4.9 l'heuristique va extraire la structure qui correspond au menu grâce à la fonction de distance qui va identifier une distance visuelle.

Dans l'exemple ci-dessous le nœud identifié par *header-nav* ne partage pas les mêmes propriétés d'arrière-plan.

Listing 4.9 – pattern eclipse.fr

```
<body>
 <div style="background:none" id="header-menu">
        <div id="header-nav" style="background:...."></div>
        <div id="header-utils"></div>
 </div>
</body>
```

règle 9 : si la taille calculée du plus grand enfant visible est plus petite qu'un seuil prédéfini alors on extrait ce nœud comme une structure logique.

Dans l'exemple 4.10, si les concepteurs respectaient les spécifications de HTML 4, ils auraient annoté le contenu comme une collection logique avec la balise *ul* (indiquant une liste). Pour détecter ce genre de configuration on peut vérifier la taille du plus grand des enfants. Si celui-ci est faible alors on peut en déduire que les enfants de ce groupe de nœud n'introduisent pas de nouvelles structures logiques.

Listing 4.10 – pattern legibase.htm

```
<div class="blockMenuGauche taille-relatif=5">
  <div class="menuGaucheInact taille-relatif=5"><a href=""></a></div
  <div class="menuGaucheInact taille-relatif=5"><a href=""></a></div
  <div class="menuGaucheInact taille-relatif=5"><a href=""></a></div
  <div class="menuGaucheInact taille-relatif=5"><a href=""></a></div
</div>
```

règle 10 : par défaut nous parcourons le nœud.

Sur les treize heuristiques de [3] nous avons adapté les règles 1, 2, 3, 8, et 9 d'après les concepts spécifiés plus haut, simplement en remplaçant les concepts de nœuds textes et nœuds textes virtuels par nos concept de nœuds de contenus et nœuds de contenus virtuels. D'après nos testes, les autres heuristiques présentées par les auteurs n'avaient pas d'impacts sur le découpage.

4.3 Annotation de structures

Dans la section 4.2 nous présentons une méthode permettant de partitionner une page. Le résultat est l'obtention d'une structure hiérarchique où chaque noeud correspond à un regroupement de nœuds apportant ensemble une même sémantique. Une partie des noeuds de cette structure est directement aliénable avec notre méta-modèle. Cependant une partie ne l'est pas, c'est le cas des méta-classes : Article, Region, Navigation, Complementary, Banner, Main, ContentInfo,*etc.*. Nous introduisons quelques pistes pour annoter les structures extraites de manière à les assimiler aux éléments du méta-modèle (cette partie n'est pas encore implémentée). Pour annoter les nœuds nous proposons d'utiliser l'approche basée sur les propriétés fonctionnelles proposée par [4] (*cf.* section 3.3). D'après les fonctionnalités associées à chaque nœud (à l'aide d'un arbre de décision) nous construisons des fonctions pour découvrir la sémantique des objets. Afin de faciliter cette découverte nous rajoutons une propriété sur la localisation d'une structure dans la page que nous nommerons "position". En effet, nous supposons que la position des objets dans la page est discriminante car les concepteurs suivent plus ou moins un schéma standard dans la présentation des pages. L'information sur la localisation de l'objet dans la page est facilement fournie par le moteur de rendu graphique du navigateur. Les valeurs possibles que nous avons choisies sont celles proposées par l'article sur la segmentation par pattern [11] (*cf.* section 3.2.2) : Bannière, Pied de page, Menu lat. gauche, Menu lat. droit. En effet, d'après les résultats du tableau 3.1, ce pattern de présentation est très représentatif des pages web.

Navigation On peut décrire un menu de navigation comme un objet composite dont la majorité des enfants de la racine sont des objets de navigation.

Banner En analysant visuellement une bannière on comprend qu'elle donne des renseignements généraux sur la page que l'on est en train de consulter. Elle se localise en haut de la page et se compose le plus souvent :

- d'un logo qui donne l'identité visuelle du site
- d'un titre introduisant le domaine (adresse du site) ou résumant le contenu principal de la page
- d'un menu de navigation qui peut être vu comme la description de l'ossature du site (présente les différentes pages du site web)
- d'un formulaire soit pour rechercher un contenu dans le site web soit pour se connecter à son compte utilisateur

Sur la base de ces éléments on peut décrire une bannière comme étant une composition d'objets :

- d'objets simples de type *Informative* qui peuvent avoir des propriétés de sémantème différentes : une image pour le logo, un titre
- d'objets composites de type *Navigation* dont les éléments de navigation pointent sur le même domaine (espace de nom) que le site web
- d'objets de type *Interactif* avec une propriété de sémantème formulaire

Elle doit avoir une propriété de position égale à *Bannière*.

ContentInfo La méta-classe *ContentInfo* fournit un contenu relatif à la page. Elle correspond à un ensemble de liens de navigation qui pointent sur des pages fournissant une information sur la page (*e.g* licence de publication de la page, politique de confidentialité,*etc.*). Traditionnellement ces informations sont placées en bas de page. On peut décrire une instance de *ContentInfo* comme une composition d'objets de navigation avec la propriété de position égale à *Pied de page*.

Main Par vacuité de valeur pour la propriété position, c'est-à-dire que l'objet n'a pas au moins l'une des valeurs définies (bannière, pied de page, menu lat. gauche, menu lat. droit) on qualifiera la structure encapsulant les nœuds n'ayant pas de valeur position comme instance de la méta-classe *Main*.

Discussion

Les concepteurs de page web suivent implicitement des standards de présentation dans les pages. On remarque que les présentations sont plus ou moins les mêmes. Cependant chaque concepteur intègre ses propres variantes. L'inconvénient dans la conception de nos fonctions est qu'elles manquent de souplesse. L'auteur de [9] se retrouve dans une configuration similaire. Dans la conception des architectures logicielles, les concepteurs se basent sur des patterns architecturaux, des solutions standards à des problèmes de conception logiciel connus, pour construire l'architecture de leur logiciel. Les architectures logicielles conçues à partir de ces patterns en sont des variantes. [9] souhaite reconnaître des motifs de conception architecturaux dans les logiciels, pour cela il propose d'exploiter une approche basée sur la programmation par contraintes pour détecter des motifs de manière plus flexible. Les motifs sont décrits par un ensemble de contraintes, si aucun motif n'est détecté alors certaines contraintes sont relâchées. Ce mécanisme de relaxation des contraintes permet de supporter une identification complète ou approchée du motif recherché. La problématique de [9] est vraiment très proche de ce que l'on souhaite faire, nous pensons

que cette piste serait intéressante à expérimenter dans notre processus d'annotation.

Chapitre 5

Expérimentations et résultats

Dans cette partie du rapport nous présentons les résultats de notre algorithme basé sur les propriétés de mise en forme (*cf.* section 4.2). Nous illustrons le déroulement de l'algorithme de manière visuelle sur une page web et dans la structure DOM de celle-ci. Nous nous intéressons à la performance de notre algorithme et aux points devant être améliorés. Notre algorithme est implémenté par un script Javascript qui est injecté à la fin du chargement d'une page sur le navigateur web. L'injection du script se fait par l'intermédiaire du pluging Greasemonkey installé sur notre navigateur web. L'expérimentation est réalisée dans un navigateur Firefox version 21.0.

5.1 Extraction

La figure 5.1a présente une page passée en entrée de notre algorithme. La structure de cette page se décrit comme suit : une bannière, un menu de navigation principal, un menu de navigation secondaire (marge latérale gauche), un contenu principal, un contenu publicitaire (marge latérale droite). La figure 5.1b montre visuellement le découpage effectué dans le DOM par rapport aux éléments de la page affichés dans le navigateur web. On peut constater que les structures extraites à la première itération correspondent bien à la structuration logique de la page décrite précédemment.

La figure 5.2 détaille de manière plus spécifique le découpage du DOM de la page en se focalisant sur la bannière. On constate que le découpage effectué est bien conforme à la sémantique contenu (*cf.* figure 5.2). La figure 5.2c schématise le DOM de la page web. Les nœuds contenant trois points de suspensions correspondent à une application successive de la règle 2. Les structures S1, S1_2, S1_3, S1_3_1 et

S1_3_2 sont découvertes par la règle 9 et la structure S1_1 est découverte par la règle 3.

5.2 Analyse

Pour analyser nos résultats, nous utilisons deux métriques. Une métrique de rappel et une métrique de précision. Ce sont des mesures standards dans le domaine de la recherche d'information pour mesurer l'efficacité d'un algorithme de recherche. Nous souhaitons mesurer l'efficacité de la recherche de structure dans une page web par notre méthode.

La métrique de rappel sera définie par le nombre de structures pertinentes trouvées au regard du nombre de structures pertinentes que possède la page. Nous mesurons cette métrique à chaque itération. Si la métrique est égale à un alors nous arrêtons les itérations car toutes les structures ont été trouvées. Cela va permettre de voir au bout de combien d'itération notre algorithme est efficace.

Dans la page de Berger-Levrault, nous mesurons l'efficacité de notre méthode pour détecter ses structures principales. Nous sélectionnons les structures suivantes comme échantillon :

- Article
- Banner
- ContentInfo
- Main
- Navigation

La métrique de précision sera définie par le nombre des structures pertinentes trouvées sur le nombre de structures totales retournées. Nous mesurons cette métrique pour chacune des itérations nécessaires à la découverte de toutes les structures attendues. Cette mesure va impacter la rapidité de l'algorithme d'annotation si notre algorithme renvoi des structures non pertinentes.

Si l'on regarde les figures 5.2b et 5.2a. A la première itération l'algorithme découvre la structure 3 (menu lat. gauche), la structure 2 (menu lat. droit) et la structure 4 (contenu principal) étant respectivement deux instances de *Navigation* et une instance de *Main*. A la seconde itération l'algorithme découvre la structure 1 et la structure 6 étant respectivement une instance de la classe *Banner* et une instance de la classe *ContentInfo*. A la troisième itération l'algorithme découvre la structure 1_3_1 correspondant à une instance de la classe *Navigation*. Pour finir la dernière itération découvre les instances de la classe *Article*.

5.3 Discussion

Au vu des premiers résultats le nombre d'itérations peut sembler conséquent cependant il est difficile d'améliorer l'efficacité de l'algorithme pour une page sans dégrader le résultat de l'application de notre méthode sur d'autres pages. Nous remarquons que la précision de notre algorithme est très faible au delà du premier tour. Le bruit correspond aux structures découvertes n'appartenant pas à la classe des échantillons. La mesure sanctionne donc sévèrement notre algorithme alors qu'il renvoie tout de même des structures correspondant à des classes de notre méta-modèle. D'autres points posent problème dans notre algorithme. En effet dans certaines configurations notre algorithme n'extrait pas les instances de la classe *Article*. Si l'on regarde le listing 5.1 l'implémentation actuelle de notre algorithme ne permet pas de regrouper le corps de l'article avec son titre. Notre algorithme extrait quatre structures logiques au lieu d'en extraire seulement deux. Une piste serait de gérer ce genre de cas par le processus d'annotation. Nous pourrions envisager d'annoter un nœud titre suivit d'un nœud texte comme un article.

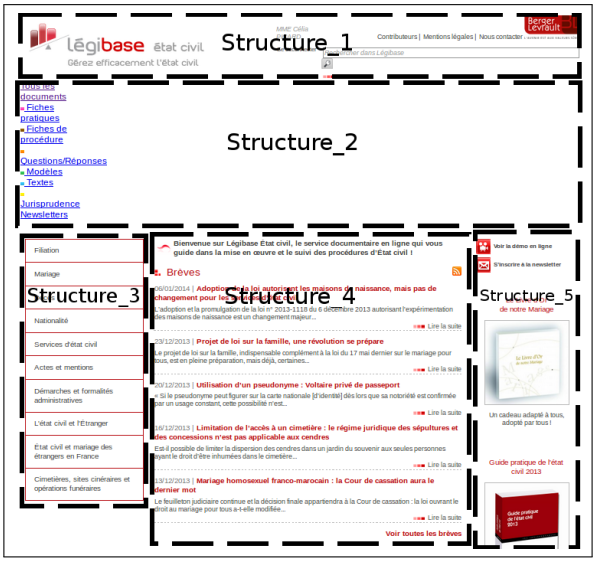
Listing 5.1 – pattern fr.wikipedia.org

```
<div class="Main">
  <div id="article-1-titre"></div>
  <div id="article-1-corps"></div>

  <div id="article-2-titre"></div>
  <div id="article-2-corps"></div>

  <div id="article-3-titre"></div>
  <div id="article-3-corps"></div>
</div>
```

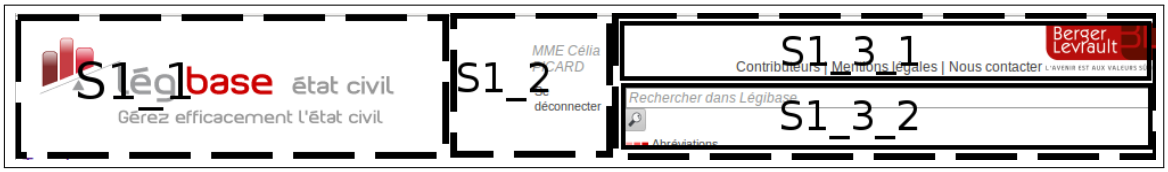
(a) Avant segmentation



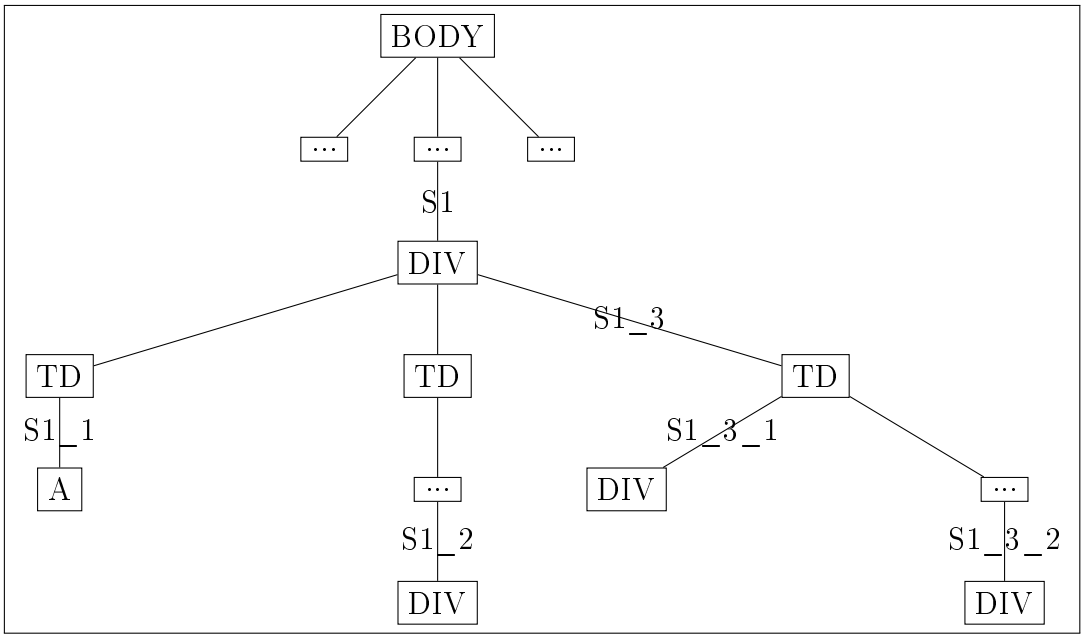
(b) Après segmentation

FIGURE 5.1 – Application du processus de segmentation sur une page web de Berger-Levrault

(a) Avant segmentation



(b) Après segmentation



(c) DOM segmentation

FIGURE 5.2 – Détail du découpage

Classe de structure	Itération 1	Itération 2	Itération 3	Itération 4
Article	0	0	0	5
Banner	1	1	1	1
ContentInfo	0	1	1	1
Main	1	1	1	1
Navigation	2	2	3	
Rappel :	0.36	0.45	0.54	1

FIGURE 5.3 – Métrique de rappel

Classe de structure	Itération 1	Itération 2	Itération 3	Itération 4
Article	0	0	0	5
Banner	1	1	1	1
ContentInfo	0	1	1	1
Main	0	1	1	1
Navigation	2	2	3	3
Bruit	0	5	37	7
Précision :	1	0.5	0.66	0.61

FIGURE 5.4 – Métrique de précision

Chapitre 6

Conclusion

En conclusion, lors de ce stage nous avons proposé un méta-modèle de page web en réponse à la diversité de représentation des éléments conceptuels dans une page. Ce méta-modèle possède un haut niveau d'expressivité comparé aux langages de publication des pages web couramment utilisés. Il répond aux besoins d'expressivité identifiés dans le cadre d'expression de préférences sur l'accessibilité. Il devra cependant être étendu à d'autres aspects que celui de la présentation d'une page web comme la navigation dans une page et ses aspects dynamiques.

Nous avons également adapté et implémenté une méthode basée sur l'analyse des propriétés de mise en forme pour extraire les différentes structures logiques d'une page web. Les premiers retours nous permettent de partiellement valider notre hypothèse de départ sur la possibilité d'extraire la structure des pages d'après leur propriété de mise en forme. En effet notre approche doit être expérimentée sur un jeu de données plus larges et représentatifs que celui réalisé. En premier lieu pour valider notre approche et en second lieu pour mettre en relation nos résultats avec d'autres travaux.

Nous avons proposé des pistes pour annoter les structures extraites d'une page. Malheureusement par manque de temps nous n'avons pas pu les implémenter pour les tester. La suite des travaux devrait se concentrer sur l'implémentation et l'évaluation de ces dernières.

En résumé ce travail était axé sur la compréhension de la structure d'une page afin d'en comprendre le contenu. Nous avons mis en évidence que la propriété intrinsèque au monde du web est sa grande liberté et diversité dans ses moyens d'expression que l'on retrouve dans la conception des pages web. Les langages du web pour supporter cette diversité et survivre dans ce milieu limitent leur sémantique et intègrent une composante syntaxique très souple. Le point délicat de notre approche était de

proposer une solution capable de s'adapter à cette diversité de conception. Dans l'environnement technologique actuelle il n'est pas possible de prendre en compte cet aspect de manière sûre. La diversité donnera toujours lieu à une exception qui fera échouer l'approche mise en place. De nouveaux langages comme RDF intègrent eux une couche réflexive qui permet d'étendre de manière élégante leur expressivité. L'essor de ce genre de technologie devrait permettre d'améliorer significativement la compréhension du contenu de manière automatique et plus simple qu'actuellement pour rendre le web bien plus accessible.

Bibliographie

[1] Yoann Bonavero, Marianne Huchard, and Michel Meynard. Web page personalization to improve e-accessibility for visually impaired people. In *WEB 2014, The Second International Conference on Building and Exploring Web Based Environments*, pages 40–45, 2014.

[2] Yoann Bonnavero. *Une approche basée sur les préférences et la satisfaction de contraintes pour améliorer l accessibilité pour les personnes déficientes visuelles.* PhD thesis, 2013.

[3] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting content structure for web pages based on visual representation. In *Web Technologies and Applications*, pages 406–417. Springer, 2003.

[4] Jinlin Chen, Baoyao Zhou, Jin Shi, Hongjiang Zhang, and Qiu Fengwu. Function-based object model towards website adaptation. In *Proceedings of the 10th international conference on World Wide Web*, pages 587–596. ACM, 2001.

[5] World Wide Web Consortium et al. HTML 4.01 specification. http ://www.w3.org/TR/REC-html40/, 1999.

[6] World Wide Web Consortium et al. Cascading Style Sheets. http ://www.w3.org/Style/CSS/, 2010.

[7] World Wide Web Consortium et al. Accessible Rich Internet Applications 1.0. http ://www.w3.org/WAI/intro/aria, 2014.

[8] World Wide Web Consortium et al. HTML 5 Specification. http ://www.w3.org/TR/html5/, 2014.

[9] Yann-Gaël Guéhéneuc. *Un cadre pour la traçabilité des motifs de conception.* PhD thesis, PhD thesis, Ecole des Mines de Nantes, 2003.

[10] Christian Kohlschütter and Wolfgang Nejdl. A densitometric approach to web page segmentation. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1173–1182. ACM, 2008.

[11] Milos Kovacevic, Michelangelo Diligenti, Marco Gori, and Veljko Milutinovic. Recognition of common areas in a web page using visual information : a possible application in a page classification. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 250–257. IEEE, 2002.

[12] Kuo-Chung Tai. The tree-to-tree correction problem. *Journal of the ACM (JACM)*, 26(3) :422–433, 1979.

[13] Karane Vieira, Altigran S da Silva, Nick Pinto, Edleno S de Moura, Joao Cavalcanti, and Juliana Freire. A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 258–267. ACM, 2006.

Appendices

.1 Méta-modèle de Contenu

Widget Élément graphique dans une page web (bouton, liste déroulant, tableau, *etc.*). Il peut définir des éléments et un contenu interactif (*e.g* formulaire d'inscription, barre de recherche, fils d'actualité).

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Role
Sous Classe	Input, Command
Alignement HTML	

Input L'ensemble des éléments permettant des entrées utilisateurs.

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Widget
Sous Classe	Checkbox, Option, Select, Textbox
Alignement HTML	

Option Elément sélectionnable dans une liste.

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Input
Sous Classe	Radio
Alignement HTML	<option>

Select Élément permettant à l'utilisateur de faire des sélections parmi un ensemble de choix.

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Composite, Group, Input
Sous Classe	Combobox, Listbox, RadioGroup
Alignement HTML	

Command Élément qui exécute des actions mais ne reçoit pas d'informations en entrée.

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Widget
Sous Classe	Button, Link
Alignement HTML	

Button Élément graphique qui déclenche une action. Typiquement un bouton de validation d'un formulaire.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Command
Sous Classe	
Alignement HTML	<button>

Link Définit une référence interactive vers une ressource interne ou externe à la page. Les navigateurs implémentent un comportement de navigation. Par exemple, une navigation vers une ressource interne peut être dans une page se déplacer de l'élément d'en-tête à l'élément de pied de page. Une navigation externe à la page peut être un changement de page.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Command
Sous Classe	
Alignement HTML	<a>, <link>

Checkbox Indique qu'une instance est cochable. Trois valeurs sont possibles (vrai-faux-mixte), indiquant si l'élément est coché ou non. La valeur mixte est utilisée dans le contexte d'un groupe d'instance de type *checkbox.* Par exemple, lorsqu'il y a au moins un élément coché et un non coché (*e.g* figure 1).

Caractéristiques	Valeur
Abstrait	Non

Super Classe	Input
Sous Classe	Radiobox
Propriété	checked [vraie, faux, mixte]
Alignement HTML	<input type= 'checkbox' />

```
<input type="checkbox">I have a bike<br>
<input type="checkbox">I have a car
```

FIGURE 1 – Exemple de checkbox

Radio Radio est une instance cochable. Il fait toujours partie d'une liste d'éléments d'au moins deux éléments. Il présente la contrainte que l'on ne peut sélectionner qu'un seul élément parmi la liste de choix auquel il appartient (*e.g* figure 2).

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Checkbox, Option
Sous Classe	
Alignement HTML	☐

```
<input type="radio">Male<br>
<input type="radio">Female
```

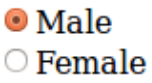


FIGURE 2 – Exemple de radio

RadioGroup C'est une collection logique d'éléments Radio. Dans HTML, la collection logique est exprimée par l'attribut name (*eg* figure 3)

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Select
Sous Classe	
Alignement HTML	

```
<input type="radio" name="vin">rouge
<input type="radio" name="vin">blanc
<input type="radio" name="vin">rose
```

FIGURE 3 – Exemple de radiogroup

Listbox Wigdet qui permet de sélectionner un ou plusieurs éléments dans une liste de choix (*e.g* figure 4).

Caractéristiques	Valeur
Abstrait	Non
Super Classe	List, Select
Sous Classe	
Alignement HTML	<SELECT>

Combobox Élément qui permet de remplir un champ texte selon des options présentées dans une liste déroulante.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Select
Sous Classe	
Alignement HTML	*cf.* figure 5

```
<select>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```

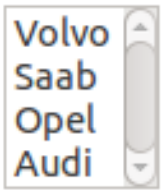


FIGURE 4 – Exemple de select

```
<input type=text list=browsers >
<datalist id=browsers >
   <option> Google
   <option> IE9
   <option> Firefox
</datalist>
```

FIGURE 5 – Exemple de combobox

Structure Éléments de structuration dans une page web. Ce sont l'ensemble des éléments qui permettent d'organiser le contenu dans une page de manière logique.

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Roletype
Sous Classe	Section, Sectionhead, Separator, Texte
Alignement HTML	

List Les listes contiennent des éléments dont le rôle est *listitem* ou des éléments dont le rôle est *group* qui contiennent eux-mêmes des éléments *listitem* (*e.g* figure 7).

	Valeur
Abstrait	Non
Super Classe	région
Sous Classe	Listbox, Ol, Ul
Alignement HTML	

Ul (unorder list) Liste non ordonnée d'éléments (*e.g* figure 6).

Caractéristiques	Valeur

```
<ul>
  <li>Cafe</li>
  <li>Thé</li>
  <li>Lait</li>
</ul>
```

FIGURE 6 – Exemple de liste non-ordonnée

OL(order list) Liste ordonnée d'éléments

Caractéristiques	Valeur
Abstrait	Non
Super Classe	List
Sous Classe	
Alignement HTML	

```
<ol>
  <li>Cafe</li>
  <li>Thé</li>
  <li>Lait</li>
</ol>
```

1. Cafe
2. Thé
3. Lait

FIGURE 7 – Exemple de liste ordonnée

Text Éléments qui définissent un état logique (sémantique) d'un texte, en opposition à un état physique (mise en forme). Elle recense donc les éléments de HTML qui apportent un état logique. On exclut donc les éléments de mise en forme tels que <b> (bolt) qui traduisent un état physique (mise en forme) mais seront exprimés dans le méta-modèle de CSS.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Structure
Sous Classe	Emphase, Abbreviation, Strong, Cite
Alignement HTML	

Emphase Mise en relief d'une partie du texte. Elle est généralement utilisée pour mettre en évidence un résumé d'article.

Abbreviation Définit une abréviation.

```
<p>Tony Blair est le premier
ministre de
la <abbr title="Grande-Bretagne">GB</abbr></p>
```

Strong Mot important dans un texte.

```
<p>Avant de faire le truc X
il est <strong>nécessaire</strong> de faire le truc
Y avant.</p>
```

Cite Elément de citation.

```
Ce référerer à la
norme <cite>[ISO-0000]</cite>
```

Section Définit une section comme une unité de confinement structurelle dans une page. On spécifie dans notre méta-modèle que les éléments héritant de section définissent des limites au contenu qu'il englobe. Ils fournissent un environnement contextuel, c'est-à-dire une portée sémantique aux éléments. Par exemple, les éléments de titre se rapportent à l'élément de section qui les ont introduits.

Caractéristiques	Valeur
isoler Abstrait	Oui
Super Classe	Structure
Sous Classe	Group, Region, Paragraph, Tablecell, Listitem
Alignement HTML	

ListItem Un élément dans une liste. Il est contenu dans une *listitem*.

Caractéristique	Valeur
Abstrait	Non
Super Classe	Section
Sous Classe	
Alignement HTML	

Group Élément regroupant une collection logique d'éléments.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Section
Sous Classe	Select
Alignement HTML	<fieldset>

Region Un groupement thématique dans une page. Éléments d'information sur une même thématique : représente une section générique d'un document.

Caractéristique	Valeur
Abstrait	Non
Super Classe	Section
Sous Classe	Article, Landmark, List
Alignement HTML	<section>

Article Contenu autonome dans une page

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Region
Alignement HTML	<article>

Paragraphe Élément ajouté au méta-modèle de ARIA. Définit une composition d'éléments textuels comme étant un paragraphe dans une page.

Caractéristique	Valeur
Abstrait	Non
Super Classe	Section
Sous Classe	
Alignement HTML	<p>

Tablecell Cellule d'une élément *Table*.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Section
Sous Classe	
Alignement HTML	<td>

Rowheader Une cellule contenant des informations d'entête pour une ligne de *Table*.

Caractéristique	Valeur
Abstrait	Non
Super Classe	Section
Sous Classe	
Alignement HTML	<th>

Tablerow Une ligne dans une *Table*.

Caractéristique	Valeur
Abstrait	Non
Super Classe	Group
Sous Classe	
Alignement HTML	<tr>

Table Élément qui contient des données tabulaires organisées en ligne et colonne.

Caractéristique	Valeur
Abstrait	Non
Super Classe	Region
Sous Classe	
Alignement HTML	<table>

Separator Élément qui marque une division dans le contenu d'une section. Il permet de mieux signaler les contenus sémantiquement différents. Ce sont des séparateurs visuels (lignes de pixels vides horizontales ou verticales entre deux éléments).

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Structure
Sous Classe	
Alignement HTML	<hr>

SectionHead Élément qui résume ou décrit brièvement le sujet introduit par une section.

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Structure
Sous Classe	Heading
Alignement HTML	

Heading Définit un élément titre

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Sectionhead
Sous Classe	
Alignement HTML	<H1,2,3,4,5,6>

Landmark Éléments structurels courants dans une page web.

Caractéristiques	Valeur
Abstrait	Oui
Super Classe	Region
Sous Classe	Banner, Main, Form, Navigation, Complementary, ContentInfo, Application
Alignement HTML	

Banner Pour faire l'analogie avec l'entête d'un document, on parle de bannière pour une page web.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Landmark
Sous Classe	
Alignement HTML	<header>

Application Contenu applicatif dans la page.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Landmark
Sous Classe	Audio, Video
Alignement HTML	<header>

Complementary Région d'un document conçue comme étant complémentaire au contenu du document auquel il est associé.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Landmark
Sous Classe	
Alignement HTML	<ASIDE>

ContentInfo Région d'un document contenant des informations sur celui-ci. Par exemple le copyright associé à un document.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Landmark
Sous Classe	
Alignement HTML	

Form Élément qui contenant une collection d'éléments formant un formulaire. Les éléments sont généralement une collection de *command, input* qui permettent une interaction avec l'utilisateur. Les interactions permettent d'envoyer des informations à un agent en vue d'un traitement.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Landmark
Sous Classe	
Alignement HTML	<form>

Main Le contenu principal dans une page.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Landmark
Sous Classe	
Alignement HTML	

Navigation Région contenant une collection de liens navigables vers des ressources internes ou externes. Par exemple le menu de navigation d'une page web.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Landmark
Sous Classe	
Alignement HTML	<nav>

.2 Méta-modèle de mise en forme

Méta-classe Box La méta-classe *Box* décrit les propriétés de positionnement des éléments de contenu. Les différentes propriétés sont :

- display : sert à définir le schéma de positionnement appliqué à la boîte. Les deux principaux étant *inline* et *block*. inline positionne les éléments sur la même ligne alors que *block* positionne les éléments les un sous les autres.
- margin (top, left, right, bottom) : spécifie l'espacement du bord extérieur de la boîte.

Méta-classe Style La méta-classe *Style* décrit les boîtes rectangulaires qui sont générées pour les éléments de l'arbre du document et leurs propriétés de positionnement. Les différentes propriétés sont :

- padding : l'aire d'espacement (*padding*)
- border-width : épaisseur de bordure
- border-style : style de la bordure
- border-color : la couleur de bordure
- border-[color, image] : arrière-plan

Méta-classe Text La méta-classe *Text* décrit la représentation visuelle des caractères, mots et paragraphes contenus dans une boîte. Les différentes propriétés sont :

- text-indent : décrit un alinéa
- text-align : décrit un alignement. Exemple de valeur possible : alignement de texte à gauche, droite, centré, *ect.*
- decoration : décrit un trait en-dessous, trait au-dessus, rayure et clignotement
- text-shadow : décrit des effets d'ombrage appliqués au texte
- letter-spacing : décrit l'espacement entre les mots
- word-spacing : décrit l'espacement entre les mots
- text-transform : décrit les effets de capitalisation dans le texte. Par exemple la valeur *uppercase* définit que les lettres de chaque mot soient en majuscule, *lowercase* décrit l'inverse.
- color : décrit la couleur du texte

Méta-classe Police La méta-classe *Police* décrit la représentation visuelle des caractères :

- font-family : décrit les noms de famille générique de la police du texte (*e.g new century schoolbook*)
- font-style : style de la police (*e.g italic*)
- font-weight : décrit la graisse de la police
- font-size : décrit la taille de la police

Résumé

Le web est un enjeu sociétal important, c'est une source d'information conséquente qui se doit d'être le plus accessible possible. Les personnes en situation de handicap visuel accèdent généralement à ces ressources au travers des navigateurs web équipés d'outils d'accessibilité. Ces outils ont besoin de comprendre les informations structurant une page pour la rendre plus accessible. Les langages de publication de pages web ne permettent pas d'expliciter ces différentes informations de manière explicite et standard. La littérature actuelle ne propose pas de solutions permettant de déduire efficacement les informations structurant une page. Dans ce mémoire de stage nous proposons une approche pour répondre à ce besoin d'inférence du contenu par segmentation visuelle.