Académie de Montpellier
Université Montpellier II
Sciences et Techniques du Languedoc

MÉMOIRE DE

STAGE RECHERCHE MASTER M2

effectuée 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**

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

Table des matières

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 ordinateurs : HTML. Historiquement conçu pour faciliter l'échange d'articles dans la communauté scientifique, la démocratisation du web a fait radicalement évoluer le contenu d'une page 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. 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, un article qui caractérisent 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. 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. Les effets sont que les concepteurs de pages web, grâce à la grande souplesse syntaxique du langage, conçoivent diverses structures pour décrire une information. Les structures ne respectent aucun schéma standard. L'analyse du contenu de la page est devenu impossible puisque chaque concepteur suit son propre schéma de conception.

La compréhension du contenu par un processus automatiques c'est ainsi complexifiée. Ce manque de compréhension du contenu des pages impactent 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'accessibilités destinés aux personnes en situation de handicap visuelles.

La question de recherche posée par le stage s'intéresse au moyen d'inférer, par un processus automatiques, les différentes informations structurant une page web. Pour cela nous étudierons dans un premier temps l'environnement des pages web afin de conceptualiser les différents éléments constitutifs d'une page web. Cette étude se concrétisera par la réalisation d'un méta-modèle de page web. Nous validerons ce méta-modèle en évaluant l'expressivité de celui par la modélisation des préférences d'utilisateurs sur l'accessibilité des pages web, acquises par une étude sur l'accessibilité du web. Une fois cette étape achevée nous élaborerons une méthode d'extraction des différents éléments constitutifs d'une page web qui sera complété par un processus d'annotation de ces éléments au regard du méta-modèle. La validation de l'approche consistera à vérifier la conformité des éléments extrait et annoter d'après les spécifications du méta-modèle construit.

La 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 la 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 associé élément de langage pour isoler les groupes qui structurent une information. Nous employons par la suite, structures logiques pour désigner un ensemble d'éléments qui mis en relation remplissent une fonction (*e.g* un menu, barre de recherche).

Ce rapport ce structure comme suit. La chapitre 2 présente des préférences sur l'accessibilité. Le chapitre 3 introduit tous d'abord une synthèse des langages de publication que nous avons réalisé à partir des spécifications des différents langages de publication de contenu. Nous y présentons également des approches existantes permettant d'isoler les différentes structures logiques et des moyens permettant d'associer une sémantique aux structures logiques identifiés. Dans le chapitre 4 nous présentons le travail réalisé pendant le stage. Dans la première partie du chapitre nous présentons la réalisation de notre méta-modèle de page web. Dans la seconde partie du chapitre, nous proposons l'adaption d'une méthode pour pour identifier la structure d'une page au moyen d'un algorithme de segmentation basé sur des heuristiques. Cette méthode est complété par un processus d'annotation des structures extraites. Le chapitre 5 présentons et analysons les premiers résultats et une discutons de ceux-ci. Le chapitre 6 présente la synthèse des travaux effectués et leurs perspectives.

Chapitre 2

Accessibilité

FP *Cette partie sera complétée par l'intégration de préférences utilisateurs recueillies dans le rapport de stage de Yohan*

différentes configurations peuvent rendre un contenu inaccessible, par exemple :

- des contrastes de couleur trop proche entre un texte et son arrière plan peuvent rendre ce texte illisible. Des combinaisons de couleurs doivent être évité.
- certaine région de la page ne sont pas utilises à la compréhension du contenu et surchargent visuellement cette dernière. Une telle surcharge peut rendre inaccessible les informations pertinentes. Un utilisateur peut souhaiter masquer les menus de navigation, la bannière et ne mettre en avant que les contenus informatifs.
- dans des cas basse vision, des utilisateurs vont devoir zoomer fortement sur contenu d'une page. Dans ce contexte une trop forte différence de entre la taille du texte et son titre rend le titre illisible. Un utilisateur peut exprimer une préférence sur la différence de contraste maximum qu'il peut y avoir entre un texte et ces éléments de titre

Chapitre 3

État de l'art

3.1 Langage de publication de page web

3.1.1 HTML 4

HTML 4 [3] 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 section et sous-section.

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 : CSS[4] 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.

3.1.2 HTML 5

HTML 5 [6] 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 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 :

- 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.

```
<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

3.1.3 ARIA

ARIA (Acessible Rich Internet Application) [5] est la spécification d'une Ontologie décrivant une interface graphique. Elle fournit des informations sur la structuration 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 widget (alertdialog, button, slider, scrollbar, menu, etc.)
- Document Structure Roles : décrit les structures qui organisent un document (article, definition, entête, ect.)
- Landmark Roles : décrit les régions principales d'une interface graphique (main, navigation, search, etc.)

```
<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

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, 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.

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

FIGURE 3.4 – Exemple d'attribution de rôle

```
<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).

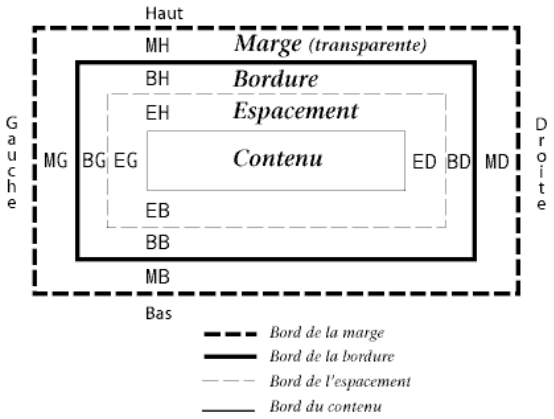

FIGURE 3.6 – Modèle de boîte

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*.

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

FIGURE 3.7 – Exemple élément en-line

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 boîte.

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 et, d'espacement et de bordure. Le couleur d'avant-plan correspond à la couleur du contenu de texte d'un élément.

Les polices CSS permet de pouvoir spécifier l'utilisation de plusieurs représentation pour les caractères textuelles : *la police*. Une liste exaustive de propriètés permettent de spécifier la police d'un élément contenu dans une boîte. On peut spécifier par exemple une famille de police (serif, sans-serif), le style de la police (italic, oblique), la taille, *ect.*

Les textes CSS définies la représentation visuelle des caractères, des caractères blancs, des mots et des paragraphes. On peut spécifié 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

3.2.1 Mapping

Un 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. Nous donnons ci-dessous une définition formelle d'un mapping (introduite par [9]).

Definition. Un Mapping M de l'arbre T1 vers l'arbre T2 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 $(i1, j1), (i2, j2) \in M$:

- i1=i2 si et seulement si j1=j2 (one-to-one condition)
- t1[i1] est à gauche de t1[i2], si est seulement si t2[j1] est à gauche de t2[j2] (préservation de l'ordre des nœuds frères)
- t1[i1] est un ancêtre de t1[i2] si et seulement si t2[j1] est un ancêtre de t2[j2] (préservation de l'ordre des ancêtres)

La première condition établie qu'une arête ne peut apparaître pas plus d'une fois dans le mapping, la seconde condition impose une préservation de l'ordre des noeuds frère et la dernière impose une relation d'héritage entre les noeuds de l'arbre. La figure 3.8 illustre un mapping entre deux arbres. De

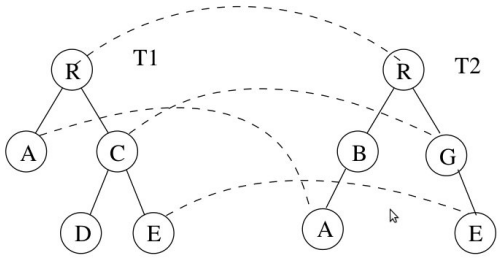

FIGURE 3.8 – Mapping entre deux arbres étiquetés et ordonnés

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 $b \to c$, où b et c sont soit un nœud, soit Λ. Alors :

- $b \to c$ 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 différente restriction sur les opérations d'édition dans un mapping, plusieurs classes de mapping se distinguent modélisant différentes applications pratiques. Notamment le mapping descendant qui restreint les opérations de suppression et d'insertion qui ne peuvent avoir lieu que sur les feuilles de l'arbre.

Definition. Un Mapping M de l'arbre T1 vers l'arbre T2 est descendant si seulement toute les paires $(i1, i2) \in M$ il y'a aussi une paire $(parent(i1), parent(i2)) \in M$

A compléter... Cette classe de mapping d'identifier les sous-structures communes entre deux arbres. Cette propriété est exploité dans une approche proposée par [10] pour identifier des structures dans une page web. L'observation des auteurs est le contenu des pages web est généré par des scripts des auteurs est que les scripts de génération de contenu qui réutilise les mêmes modèle pour structurer certain type de contenu, en particulier les bannieres, menu, et plus généralement les contenus peu important dans une page (du point de vu de l'information apporté). La démarche est la suivante, à partir d'un échantillons de page web les auteurs identifies les sous structures commune des pages au moyen d'un mapping descendant. A partir du mapping entre les pages, les auteurs construisent ainsi des patterns.

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 des objets HTML (noeuds du DOM de page web) suivant des propriétés prédéfinies. Le but étant de proposer une structure pour la page web traité.

Segmentation par pattern de page

Les auteurs de [8] proposent dans leur papier une segmentation suivant un pattern présentation défini à priori. Cette approche s'intéressent aux données cartésiennes des objets HTML 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 objets HTML suivant leurs localisations.

Sur la base d'une étude empirique d'une Les auteurs proposent un pattern de présentation de page web (*cf. figure*3.9). Le modèle construit est le suivant :

- 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)
- du centre de la page encapsulant le contenu principal de la page ((C))

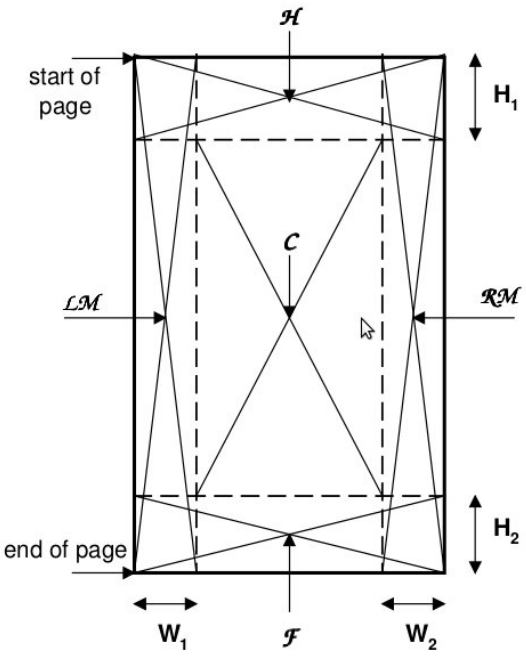

FIGURE 3.9 – Pattern de mise en forme de page web

La figure 3.9 modélise le partitionnement définit par les auteurs avec *W1* et *W2* définissant respectivement la marge contenant les menus latérales gauches et la marge contenant les menus latérales 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 un hauteur de 200px pour H et 150px pour F.

Les coordonnées des objets 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 objets HTML de la page suivant le partitionnement de page proposé ci-dessus. Ces heuristiques sont construite sur l'identification de noeuds de type *TABLE* dont la surface est incluse dans les

zones du modèle.

À partir 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 *Bon* est attribuée aux zones dont les objets HTML sont à plus de 50% correctement étiquetés, c'est à dire que par exemple les éléments classifier 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

	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%

D'après les auteurs 20% des mauvais résultats sont dû au moteur de rendu graphique utilisé qui construit une mauvaise représentation des objets HTML. Les positions de ceux-ci ne correspondent pas à la position réelle qu'ils devraient avoir. Les 80% restant, toujours selon les auteurs, devraient pouvoir être diminué en développant de meilleurs heuristiques.

Segmentation par densitométrie textuelle

Les auteurs 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 du DOM est une propriété suffisante pour identifier la structure logique d' une page.

La première étape consiste à identifier les différents segments textuelles atomique de la page. La page est vue comme une sequence d'élément caractères entrelacés de tag HTML. Certain tag HTML segmentent le flux textuelle et d'autres non. Cette décision est prise (non pas d'après la sémantique des balises) d'après le rythme des séquences de caractères dans le flux. Par exemple si le flux passe d'une sequence de caractères courtes à une sequence de caractères longue cela signale un nouveau segment. Par exemple, si on analyse une sequence de texte courte comme *<a>Accueille<a>New<a>Contact* suivit d'une sequence 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. Cette propriété correspond au ratio ci-dessous :

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

Les jetons correspondent à une séquence contingente de caractères n'étant pas des caractères d'espacement. Le nombre de ligne correspond au nombre de découpage de blocs 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 leur densité respectif. La fonction est définit 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 définit ci-dessus ne satisfait pas un certain seuil. Cela conduit à la construction de blocs de forte granularité correspondant à la structuration logique de la page.

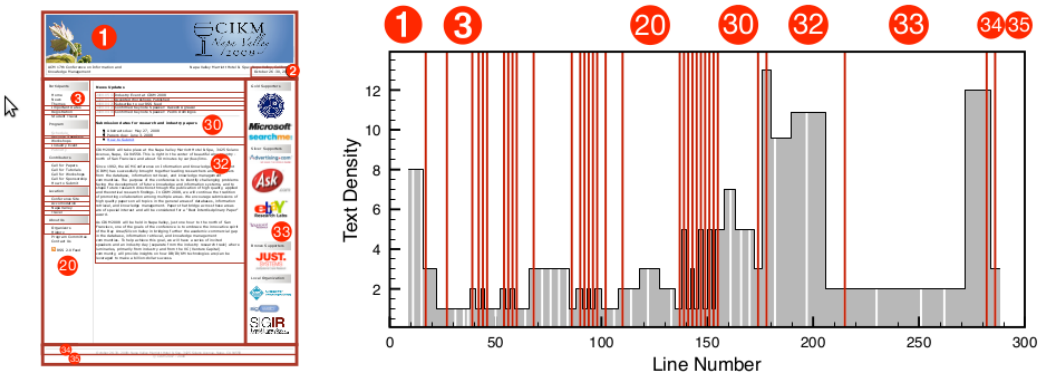

FIGURE 3.10 – Segmentation densitométrique [7]

Segmentation par indice visuel

L'approche proposée par les auteurs [1] 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 sémantiquement proches en bloc (*e.g.* figure 3.11). Le postulat est que les éléments d'une page possédant des caractéristiques de mise en forme proches, tels que la police, la couleur, la taille, sont sémantiquement proches.

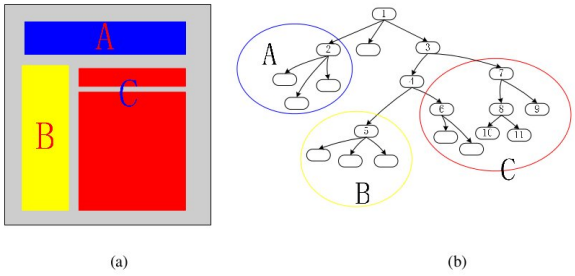

FIGURE 3.11 – Exemple de partitionnement, (a) page (b) DOM de la page [1]

L'algorithme exploite le DOM[1] de la page web. Le DOM est une API[2] pour les documents HTML (ou plus généralement XML). Il fournit une représentation arborescente d'un document et les moyens d'accéder à son contenu et sa mise en forme.

1. Document Object Model
2. Application Programming Interface

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. 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 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.

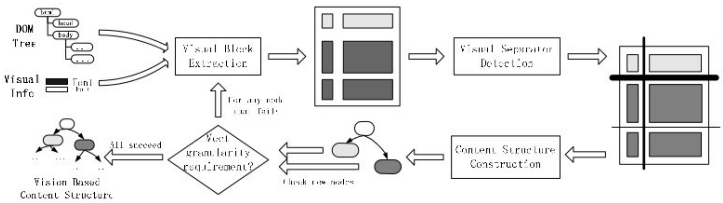

FIGURE 3.12 – Algorithme de segmentation [1]

En conclusion, l'algorithme regroupe les éléments d'une page en structure ayant une sémantique forte. Cette méthode nous permet de délimiter les différentes régions d'une page. Cependant l'inconvénient est que l'algorithme ne nous permet pas de connaître le rôle associé à chaque sous-structure isolée dans la page.

3.2.3 Synthèse

L'approche par correspondance de structure (section 3.2.1) entre les pages permet d'isoler seulement des fragments de pages, la structure globale de la page n'apparaît pas . Les approches par segmentation sont plus intéressante car elle propose un découpage de la page reflétant une struc-

ture. 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 mise en forme. La segmentation par densitométrie (section 3.2.2) est une méthode indépendant du pattern de mise en forme de la page. Le problème avec cette approche est qu'elle ne prend pas en compte les possibles écarts entre la structure décrite par HTML et le rendu finale. C'est à dire que des noeuds du DOM peuvent être proches du la structure du DOM et avec l'application de propriété de mise en forme, très éloigné avec le rendu visuel en particulier avec les propriétés de positionnement absolue décrite en section 3.1.4. La segmentation par indice visuel (section 3.2.2) qui comble les faibles des deux méthode précédentes. L'inconvénient de cette approche se situe au niveau de l'algorithme du calcule des séparateur qui présente une grande complexité. FP *Je vais rajouter la complexité du calcule des séparateurs*

3.3 Annotation de structures

3.3.1 Approche fonctionnelle

[2] propose une approche afin de découvrir la fonctionnalité des éléments composant une page web. Les différents éléments sont des objets qui apportent une information ou réalise une action. Ces objets peuvent soit être des objets basiques ou soit 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 noeuds ayant des descendants forment des objets composites. Afin de qualifier chaque noeuds les auteurs proposent un ensemble de propriété 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 ça sémantique.

La structures des objets basiques est décrit 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'esthétisme de la page (lignes, séparateurs, et les objets avec des couleurs d'arrière plan).

Hyperlien : l'objet possède t il des hyperliens ? et 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'objet basique et/ou composite. 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 même 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

– Spatio/temporelle : ordre de présentation des objets dans le temps (ordre d'apparition) et l'espace().

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 déclenche une navigation

Interaction : objets qui fournissent des interactions avec l'utilisateur

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

La classe des objets basiques est déterminée simplement par la valeur de ces propriétés. Un objet basique informatif est possède un contenu textuel important, une image (taille significatif 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 objets composites dont les enfants de la racine sont des objet de navigation parallèle (même propriété), le nombre d'enfant navigable est supérieur à un certain seuil $Hmin$ et la propriété de texte est inférieur à un seuil $Lmax$

3.3.2 Inférence du contenu informatif

Une page web possède différent contenu que l'on peut classifier comme contenu informatif et contenu non informatif. Un contenu informatif est un contenu qui apporte une information dans dans la page. Une premiere phase est le découpage de la page en bloc, ce découpage se base sur les balises table (l'approche ne traite que les site organisées avec les balises tabulaires) Pour chaque bloc les auteurs extraits les mots clés (caractéristiques des blocs)

Inférence basée sur la localisation

La vision des auteurs est que les blocs informatifs sont plutot des blocs qui se concentrent sur le centre de l'écran. Les auteurs [11] propose un algorithme pour l'extraction du contenu informatif dans une page web. L'approche simule comment un utilisateur trouve le contenu informatif dans la page. Le contenu informatif est identifié comme étant les blocs contenant le plus de contenu textuels. L'algorithme se décompose en trois étapes :

1. Extraction des blocs sémantiques dans la page. Cette partie repose sur une méthode de segmentation similaire à l'algorithme présenté en section 3.2.2.
2. Sélection des candidats. L'algorithme prend en entrée les blocs résultants de l'étape précédente. Pour chaque bloc est appliqué une fonction qui vérifie si il peut être candidat ou non à être un bloc informatif. Cette fonction s'intéresse aux propriétés textuelles, nombre d'hyperliens. Les détails du paramétrage de cette fonction ne sont pas précisés dans l'article.
3. Élection des blocs informatifs. L'algorithme prend en entrée la sortie de l'étape précédente. Dans un premier temps, pour chaque bloc est calculé une valeur signalant l'importance de son contenu informatif.

$$f(b_x) = \frac{TailleDuBloc b_x}{DistanceEntreLeCentreDeb_x EtLeCentreDelEcran}$$

Inférence basée sur la fréquence

FP *Intégration d'un article sur l'identification du contenu informatif dans une page. Propose le calcule d'une valeur d'entropie basée sur une mesure nommée TF-IDF permettant le partitionnement des structures logiques en contenu informatif ou non informatif*

3.3.3 Synthèse

Chapitre 4

Réalisation

4.1 Méta-modèle

4.1.1 Introduction

HTML 4 fournit une sémantique pour définir des noeuds de contenu (image, texte) mais pas pour des structures logiques plus élaborées (article, menu, bannière etc.). Par exemple, la figure 4.1 présente deux structures logiques extraites d'une des pages du site web *lemonde.fr* et d'une page des pages 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 pause un problème pour l'expression de préférence sur de telles constructions.

La contribution de ce méta-modèle vient apporter une composante (dimension) sémantique aux langages de publication du web. Cette sémantique s'intéresse à la correspondance entre les structures logiques définies par le concepteur d'une page web et ce que 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.

Le méta-modèle se décompose en deux parties. Une partie décrivant la sémantique des principales structures logiques (section 3.1.4) utilisées dans la conception des pages. Une seconde partie pour la descriptions de la mise en forme associée à chaque élément du méta-modèle précédent.

4.1.2 Méta-modèle HTML 4 et 5

HTML 4

Méta-modèle HTML 4 HTML 4 discrimine ces éléments de langage suivant que pendant l'affichage du flux de contenu les éléments produisent un retour à la 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 4.2 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 (<p>) génère un retour à la ligne

```
<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 4.1 – Exemple de différentes conceptions de menu avec HTML4

– 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.

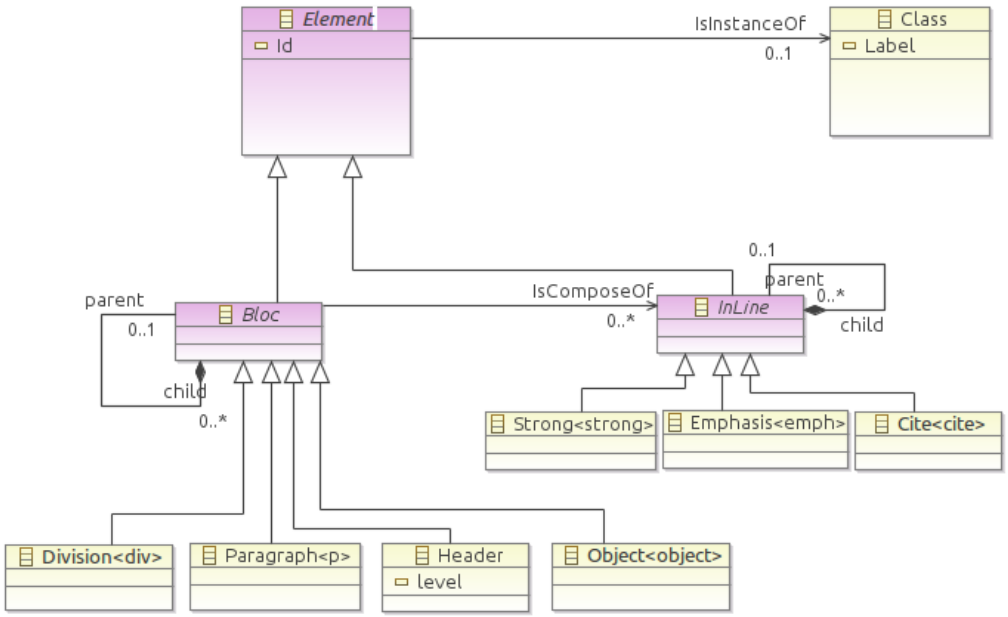

FIGURE 4.2 – Méta-modèle HTML 4

Instance HTML 4 délimitent les structures logiques de plus haut niveau à l'aide de d'éléments structurels générique (*div*) et leurs propriétés de mise en forme (bloc/inline). Si l'on instancie le premier menu de la figure 4.1, 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.

faire un schéma

HTML 5

Méta-modèle HTML 5 HTML 5 propose une classification plus élaborer qui ne se concentre plus sur les aspects de mise en forme mais sur la sémantique des différentes structures. Nous proposons dans la figure 4.3 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 définissent 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.
- La instances de la méta-classe **Interactive** définissent un contenu conçu pour un 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.

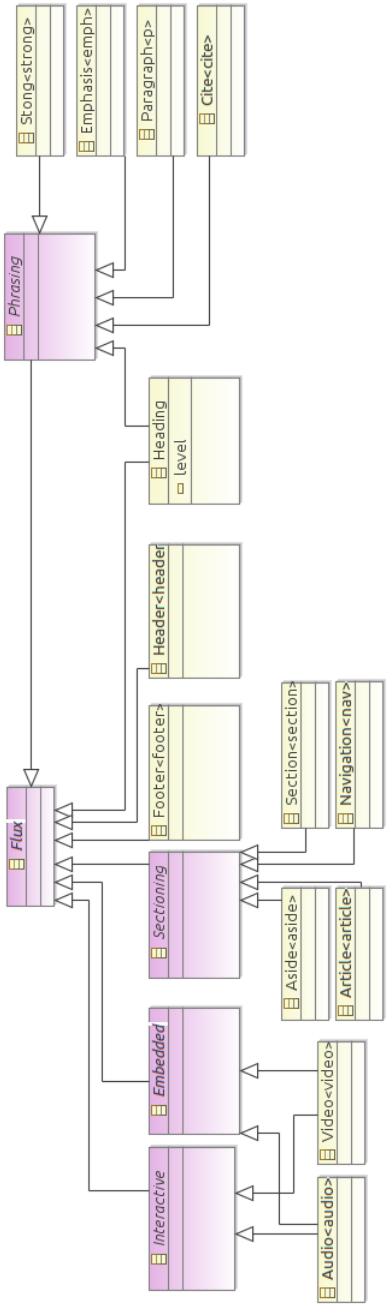

FIGURE 4.3 – Méta-modèle HTML 5

Instance 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'élément 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 4.1, 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 préférences sur l'accessibilité serait, par exemple, que la taille du texte du menu de navigation soit supérieur à 15 pixels :

```
navigation.style.size < 15px
```

faire un schéma

4.1.3 Méta-modèle

Modèle de contenu

La conception de cette partie du méta-modèle repose sur les éléments conceptuels identifiés dans la sous-section 4.1.2 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 4.4 et 4.5) construit va nous permettre de qualifier les éléments d'une page. Une première partie décrie les éléments de structuration.

Les principales méta-classes de structures sont :

- Les instances de *SECTION* peuvent contenir des instances de structure.
- 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 *LANDMARK* spécifie *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'instance de *SECTION*. Il possède une portée locale, ceux-ci synthétise le contenu de l'instance de type *SECTION* le plus proche.
- Les instances de *Texte* définissent un contenu textuelle (*e.g* une emphase, une citation, *etc.*). Elles ne peuvent contenir que des éléments textuelles ou des instances de la méta-classe *Texte*.

Un 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=...*> déclenche une action de navigation ou une balise de type <*input type=button*> déclenche l'envoie de données d'un formulaire vers un serveur.
- Les instances de *SELECT* qui permettent de faire des sélections parmi un ensemble de choix.

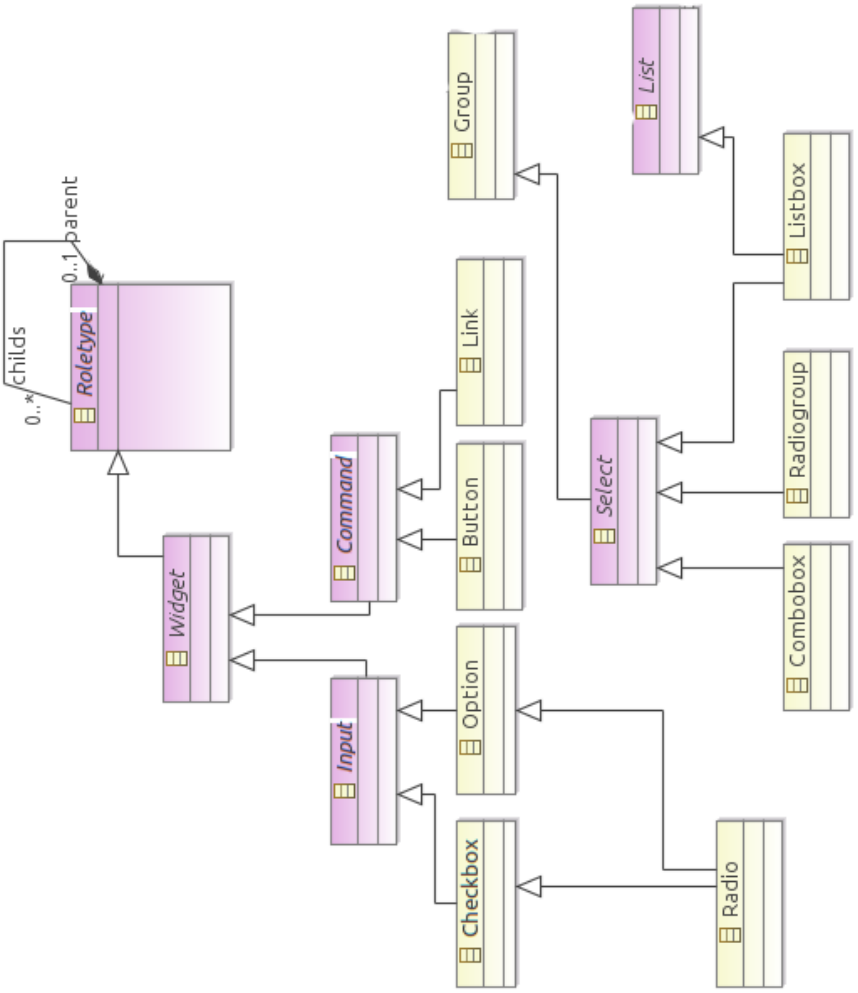

FIGURE 4.4 – Méta-modèle interaction

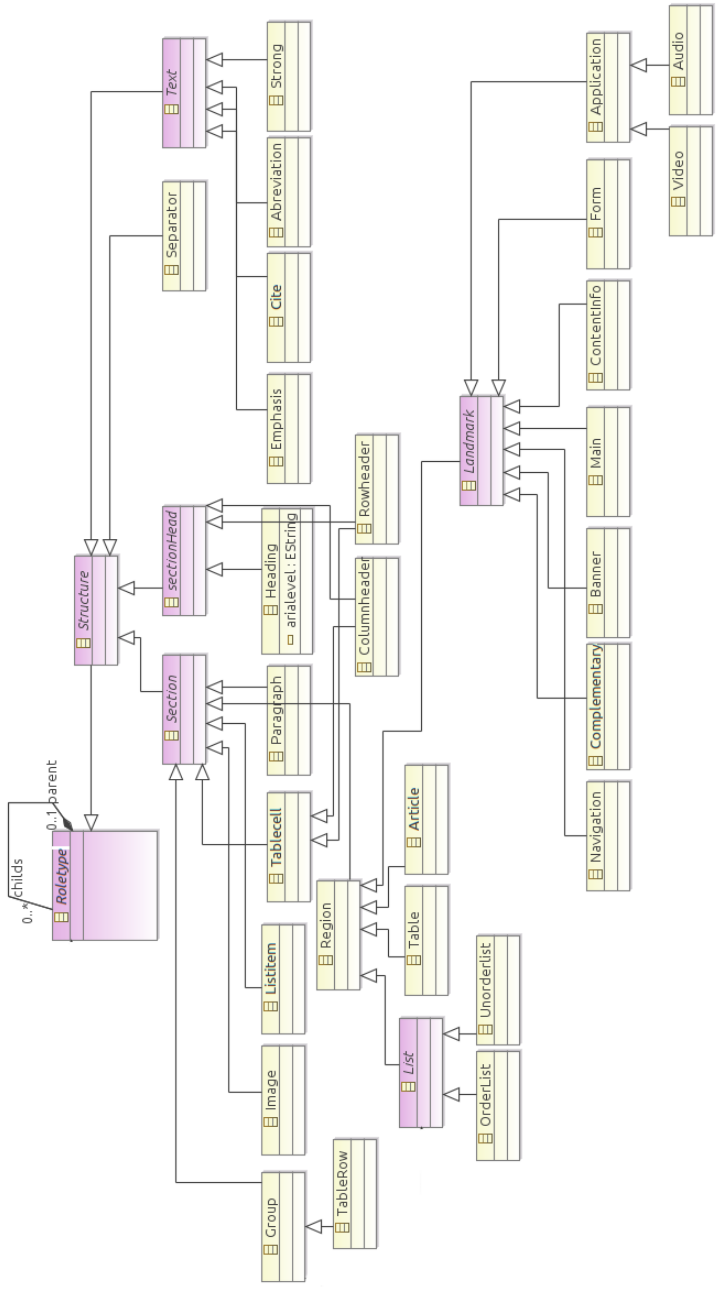

FIGURE 4.5 – Méta-modèle contenu

Modèle de mise en forme

Le méta-modèle de mise en forme (*cf.* figure 4.6) 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-classe sont :

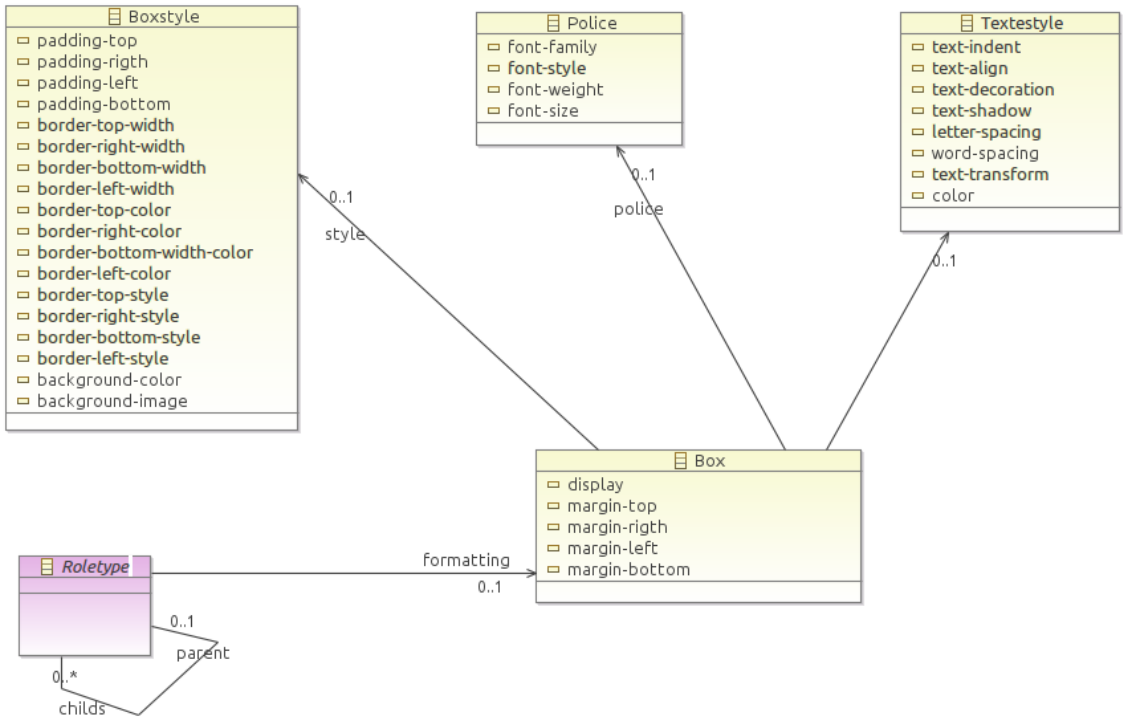

FIGURE 4.6 – Méta-modèle

- Les méta-classes *BOX* qui représentent le concept de bloc conteneur. Elles vont permettre d'exprimer des préférences sur l'accessibilité tels que l'espacement plus ou moins fort (propriété padding) avec les éléments connexes à cette dernière. 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 être intéressant, par exemple, d'éviter certain type de font qui peuvent rendre un contenu plus illisible pour certaine pathologie.

Instance du méta-modèle

4.1.4 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. On peut tout à fait exprimer des contraintes sur le contraste d'un texte

```
context Paragraph        inv :
        if self.style.color = red then
                self.style.background-color != green
```

Exprimer des sur des structures plus haut niveau

```
        Banner.style.display = hidden
```

4.2 Extraction de structures

4.2.1 Introduction

Dans cette partie nous traitons de l'extraction des structures logiques dans une page. Nous nous intéresserons à la sémantique de ses dernières dans le section **??**.

Le DOM d'une page web est constituée d'un ensemble de structures logiques. Nous employons structures logiques pour désigner un ensemble d'éléments qui mis en relation remplissent une fonction (*e.g* un menu, barre de recherche). Ces structures logiques ne sont pas annotées de manière standard dans le DOM de la page. Nous voulons dire par explicite qu'il n'y pas d'annotations standards pour identifier et qualifier une structure. Cette annotation est connu que par la concepteur au travers de l'annotation Ces structures apparaissent de manière implicites lors de l'application des propriétés de mise en page par le navigateur (couleurs d'arrière plan, typographie, etc.).

Une première difficulté provient de l'hétérogénéité du contenu. Une page définit plusieurs contenu (bannière, pied de page, article, *etc.*). Si l'on regarde le listing 4.2, dans le contexte de notre méta-modèle, on doit identifier trois structures logiques différentes : une bannière, un document, un menu. Les éléments de langages de HTML 4 ne permettent pas de signaler les différents contenu dans une page.

Listing 4.1 – Exemple contenu homogène

```
<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>
```

Listing 4.2 – Exemple 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 seconde difficulté est la diversité des structures (*cf.* section 4.1.1) complexifie la solution puisqu'on ne peut pas utiliser de méthodes basés sur la correspondance de structures.

D'autre part les concepteurs de page web ne tiennent pas toujours compte des spécifications des langages dans la création des pages. Donnons deux exemples. Dans le listing 4.3 on constate que l'étiquette *P* n'est pas utilisée pour structurer un textes en paragraphe mais pour structurer des liens de navigation en menu. Dans le listing 4.4 on voit que les étiquettes *DIV* ne sont utilisées pour indiquer une division mais pour faire de la mise en forme (en appliquant un style).

Listing 4.3 – Extrait menu lemonde.fr

```
<p>
        <a href='...'>Le Monde</a>
        <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 4.4 – Pattern page d'accueil google.fr

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

Dans notre contexte nous devons prendre en compte une notion de granularité dans les différentes structure logique que l'on peut identifier. En effet, dans notre méta-modèle, une bannière est une structure logique, mais on peut décomposée en structure logique plus fine. La bannière des pages web intègre dans leur structures des menus et des bars de recherche que nous souhaitons extraire.

Toute ces contraintes posent le cadre de l'élaboration de notre méthode d'extraction des structures logiques dans une page. Plusieurs approches sont décrites dans l'état de l'art () La segmentation par pattern est un bonne approche. On constate que dans la conception de page web que même si elle au niveau de la conception (niveau html) elles sont très différentes à l'étape de la mise en forme (publication), elles suivent un même pattern de présentation. L'inconvénient de cette approche est que la granularité des structures identifier est trop forte et que toute les pages web ne sont pas basées sur le même pattern de présentation.

Pour la conception des différentes heuristiques nous émettons l'hypothèse, comme [1], qu'elles doivent intégrer les propriétés de mise en forme associées aux neuds du DOM. Dans un premier temps nous proposons ainsi plusieurs concept pour l'élaboration d'heuristiques. Ces concepts définissent une modèle de page web intégrant les aspects de mise en forme et de structuration des éléments

4.2.2 Méthode

Description modèle de page

La méthode choisit est un processus itérative basée sur des découpages successives de la structures de la page. Depuis la racine de la structure d'une page, on va parcourir (suivant un parcours préfixé) ses noeuds. Pour chaque noeud 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 rappliquer le processus de découpage, dans le second cas on continue la parcours. Les itérations sont stoppés lorsque tous les noeuds atomiques sont identifié. Le résultat est la construction d'un arbre reflétant la composition des structures dans la page. Cette structures nous permet par la suite d'appliquer des algorithmes permettant d'inférer une sémantique aux noeuds de l'arbre.

Nous définissons ci-dessous les concepts utiles à l'élaboration des heuristiques. Nous proposons une classification des noeuds (d'après[1], une fonction de distance visuel et de calcule du poids d'une nœeuds :

Nœuds conteneur : 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.

Noeuds 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 structure pas les noeuds qu'elles encapsulent. Si les noeuds conteneurs décrivent une structure logique, les nœuds de mise en forme décrivent une structure physique.

Nœuds de contenus : l'ensemble des nœuds qui contiennent un contenu atomique. Les contenus atomiques correspondent aux éléments multimédias dans la page : texte, image, video, etc. On inclus les balises <img>, <object> et <h1-h6>

Nœuds de contenus virtuelles : 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 nœeuds de mise en forme</P>*. *P* est un nœud virtuel

Nœuds de descriptions : nœuds qui donnent des méta-informations sur une page. Par exemple, la balise *<style>* renseignent sur la feuille de style à appliquer à la page. Ce sont des balises utilisé pour renseigner le navigateur sur la page

Nœuds visibles : l'ensemble des nœuds qui sont visibles au travers du navigateur

Fonction de distance visuelle : cette fonction compare les propriétés de mise en forme associée à chaque noeuds pour déterminer si ils sont visuellement distant ou non. Par exemple, la fonction renvoie vraie si le nœuds ne partagent pas les mêmes propriétés :
- d'arrière-plan
- de bordure
- etc.

Taille du nœuds : pour chaque nœud, nous calculons taille. Le calcule de la taille prend en compte les éléments contenu par le noeud et sa position dans le DOM. $Taille(b_x) = \frac{TailleDuNoeud_x}{100} * TailleDuNoeudRacine$

Heuristiques

L'élaboration des différentes heuristiques ci-dessous reposent sur les concepts décrient précédemment.

règle 1 : Si le nœuds n'est pas un nœuds de contenus et qu'il ne possède pas de nœuds valide alors on ne traite pas ce noeud.

règle 2 : Si le nœuds possède un seul enfant valide et cette enfant n'est pas un nœuds de données alors on parcours ce nœuds

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

règle 4 : Si l'un des enfants du nœuds est un nœuds conteneur alors on parcours ce nœuds et que ça taille est inférieur à un certain seuil alors on parcours.

règle 5 : Si l'un des enfants du nœuds est étiqueté par *HR* alors on extrait les premiers nœuds valides successeurs et prédécesseurs comme des structures logiques de chaque enfant étiqueté par *HR*

règle 6 : Si la somme des enfants visibles de la surface du nœuds est supérieurs à ce dernier, alors on parcours le nœuds

règle 7 : Si le premier enfant valide du nœuds est un nœuds de titre alors on alors on extrait ce nœuds comme une structure logique.

règle 8 : Si la fonction de distance visuelle est vérifiée (renvoie vraie) pour au moins un enfant du nœuds alors on parcours ce neoud

règle 9 : Si le plus grand enfant valide est plus petit qu'un seuil prédéfini alors on extrait ce nœuds comme une structure logique.

règle 10 : par défaut nous parcourons le nœud.

Règle 1 Cette règle élimine les noeuds qui

Règle 2 La règle 2 permet de découvrir des structures logiques plus intéressantes dans une page. D'après l'exemple 4.5, cette règle va permettre d'atteindre le noeud *center* contenant la bannière de la page et la bar de recherche.

Listing 4.5 – Pattern page d'accueil google.fr

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

Règle 3 La règle 3 nous permet de récupérer les noeuds de données. Ce sont les noeuds que l'on considère comme atomique.

Listing 4.6 – 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'on reprend 4.5 après l'application de la règle 2 on va découvrir des structures logiques plus intéressantes qui sont le bannière de la page d' accueil de GOOGLE et la bar de recherche.

Règle 6 Dans 4.7 le noeud *mw-navigation* est un

Listing 4.7 – Pattern wikipedia.fr

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

Règle 7 Les balises de titre sont utilisés pour introduire 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 Dans l'exemple 4.9 l'heuristique va extraire la structure qui correspond au menu grâce à la fonction de distance qui va identifier une distance visuelle.

Listing 4.9 – pattern eclipse.fr

```
<body>
 <div id="header-menu">
        <div id="header-nav" style="background=..."></div>
        <div id="header-utils"></div>
 </div>
</body>
```

Règle 9 Dans l'exemple 4.10

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>
```

4.3 Annotation de structures

La catégorie d'un objet composite est déterminé par les propriétés majeurs

Chapitre 5

Résultats

Chapitre 6

Conclusion

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, bar 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 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 (vraie-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'élément 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
```

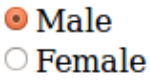

FIGURE 2 – Exemple de radio

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

Caractéritique	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 élément dans une liste de choix(*e.g* figure 4).

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

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

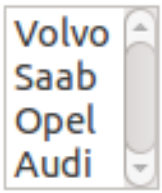

FIGURE 4 – Exemple de Select

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

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

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 même des éléments *listitem* (*e.g* figure 7).

Caractéritique	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).

Caractéritique	Valeur
Abstrait	Non
Super Classe	List
Sous Classe	
Alignement HTML	

OL(order list) Liste ordonnée d'éléments

Caractéritique	Valeur
Abstrait	Non
Super Classe	List
Sous Classe	
Alignement HTML	

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 telles 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	

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

FIGURE 5 – Exemple de Combobox

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

FIGURE 6 – Exemple de liste non-ordonnée

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

Abbreviation Définit une abreviation.

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

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

1. Cafe
2. Thé
3. Lait

FIGURE 7 – Exemple de liste ordonnée

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 contextuelle, c'est à dire une portée sémantique aux éléments. Par exemple, les éléments de titre se rapportent à l'élément de section qui l'a introduit.

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*.

Caractéritique	Valeur
Abstrait	Non
Super Classe	Section
Sous Classe	
Alignement HTML	

Group Élément regroupant une collection logique d'éléments (eg. figure **??**).

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.

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

Article Contenu autonome dans un page

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

Paragraphe Élément rajouter au méta-modèle de aria. Définit une composition d'élément textuelle comme étant un paragraphe dans une page.

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

Tablecell Cellule d'une élément *Table* (*eg.* figure **??**).

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*(*eg.* figure ??).

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

Tablerow Une ligne dans un *Table*(*eg.* figure **??**).

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

Table Élément qui contient des données tabulaires organisées en ligne et colonne (*eg.* figure **??**).

Caractéritique	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 visuelles (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 décrive 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 courant 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 conçut 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 copyrights 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 permettre d'envoyer des informations à un agent en vu d'un traitement (*cf.* figure **??**).

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

Main Le contenu principale dans une page.

Caractéristiques	Valeur
Abstrait	Non
Super Classe	Landmark
Sous Classe	
Alignement HTML	

Navigation Région contenant une collection de lien navigable vers des ressources internes ou externes. Par exemple le menu de navigation d'une page web (*cf.* figure **??**).

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 (*cf.*figure **??**). Les différentes propriétés sont :

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

Méta-classe Text La méta-classe *Text* décrit la représentation visuelle des caractères, mots et paragraphes contenu 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 appliquer 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 mots 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

Glossaire

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 [**?**]. 11

Bibliographie

[1] 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.

[2] 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.

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

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

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

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

[7] 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.

[8] 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.

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

[10] 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.

[11] Chaw Su Win and Mie Mie Su Thwin. Informative content extraction by using eifce [effective informative content extractor].