

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

1	Introduction	5
2	État de l'art	7
2.1	Extraction de structures	7
2.1.1	Mapping	7
2.1.2	Segmentation par pattern de page	7
2.1.3	Segmentation par densitométrie	8
2.1.4	Segmentation par indice visuel	9
2.1.5	Synthèse	11
2.2	Inférence de structures	11
2.2.1	Approche fonctionnelle	11
2.2.2	ITF-IDF	12
3	Langage de publication de page web	13
3.1	HTML 4	13
3.2	HTML 5	14
3.3	ARIA	15
3.4	CSS	16
4	Accessibilité	21
5	Réalisation	23
5.1	Méta-modèle	23
5.1.1	Introduction	23
5.1.2	Méta-modèle HTML 4 et 5	24
5.1.3	Méta-modèle	27
5.1.4	Modèle de contenu	27
5.2	Extraction de structures	29
5.2.1	Introduction	29
5.2.2	Méthode	31
5.3	Inférence de structures	34
6	Validation	35
7	Conclusion	37

Appendices	39
.1 Méta-modèle de Contenu	41
.2 Méta-modèle de mise en forme	54

Chapitre 1

Introduction

Chapitre 2

État de l'art

2.1 Extraction de structures

2.1.1 Mapping

2.1.2 Segmentation par pattern de page

Les auteurs de [8] proposent dans leur papier une segmentation des pages web par zone. Cette approche s'intéresse aux données cartésiennes des objets HTML. 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 leur localisation. Les auteurs proposent un pattern de page web construit suivant une observation empirique (*cf. figure 2.1*). 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))

La figure 2.1 modélise le partitionnement défini par les auteurs avec $W1$ et $W2$ définissant respectivement LF et LR . Chacune d'elle représente 30% de la largeur de la page. $H1$ et $H2$ définissent H et F avec une hauteur de 200px pour H et 150px pour F .

Les coordonnées des objets sont calculées au moyen d'un moteur de rendu graphique. Il est à noter que tel qu'il est décrit dans l'article, le moteur de rendu graphique ne prend pas en compte la surcouche CSS.

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 construites 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 classifiés comme bannière correspondent bien à une bannière. Une note de *Excellent* est attribuée aux zones dont

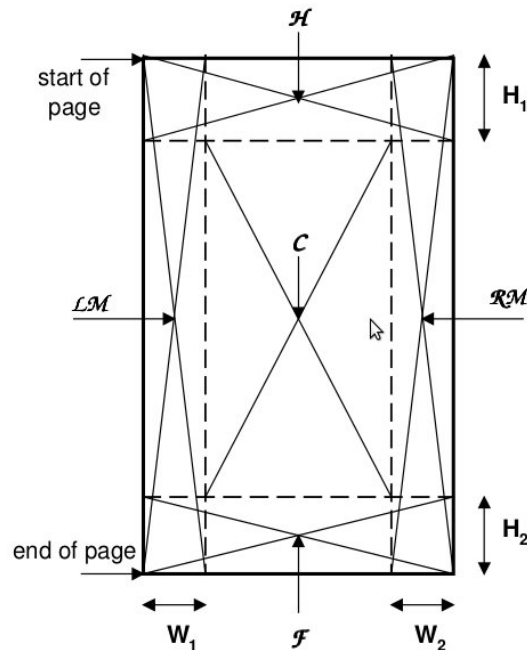


FIGURE 2.1 – Position des zones d'intérêts dans une page

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	Footer	Menu lat. gauche	Menu lat. 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.

2.1.3 Segmentation par densitométrie

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 textuelle atomique de la page. Les auteurs ne prennent pas en compte la sémantique des balises HTML dans le découpage. La page est vue comme une séquence d'élément textuelle entrelacé 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 texte dans le flux. Par exemple si le flux passe d'une séquence de texte courte à une séquence de texte longue cela signale un nouveau segment. Par exemple, si on analyse une séquence de texte courte comme $\langle a \rangle \text{Accueil} \langle /a \rangle \langle a \rangle \text{New} \langle /a \rangle \langle a \rangle \text{Contact} \langle /a \rangle$ suivit d'une séquence de texte longue comme $\langle p \rangle \text{Une séquence de texte beaucoup plus longue} \langle /p \rangle$ va donner naissance à deux nouveaux segments. Pour chaque segment (bloc) identifier par le processus ci-dessous. Les auteurs vont calculer une densité textuelle. Cette propriété correspond au ratio ci-dessous :

$$p(b_x) = \frac{\text{NombreDeMotsDans}b_x}{\text{NombreDeLignesDans}b_x}$$

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. Cette construction consiste à fusionner les blocs contingents en comparant leur densité respectif. La fonction est défini ci-dessous :

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

Ce processus 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.

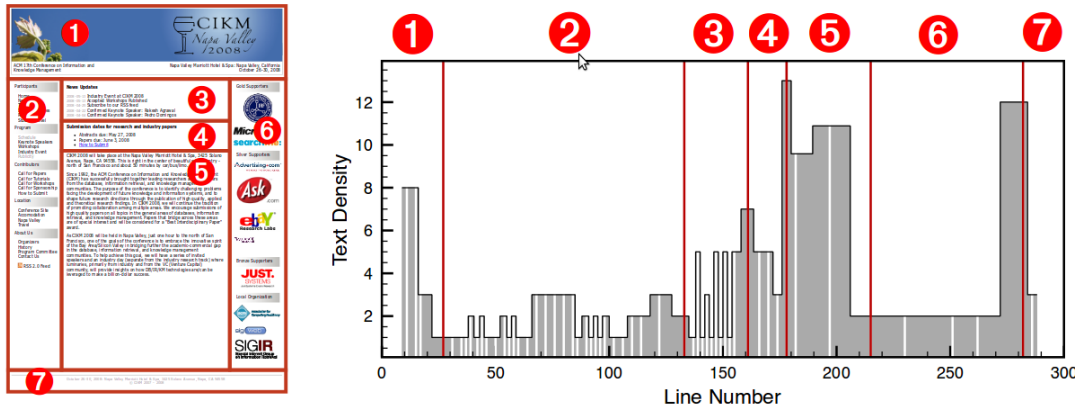


FIGURE 2.2 – Segmentation densitométrique [7]

2.1.4 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 2.3). 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.

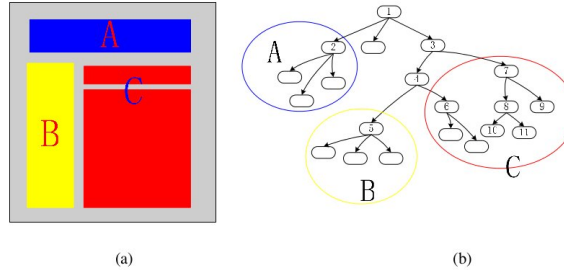


FIGURE 2.3 – Exemple de partitionnement, (a) page (b) DOM de la page [1]

L'algorithme exploite le DOM¹ de la page web. Le DOM est une API² 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.

Le processus de segmentation, figure 2.4, 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

1. Document Object Model
2. Application Programming Interface

pas supérieur au seuil de cohérence, le bloc est de nouveau proportionné. La structure finale est construite après que tous les blocs soient traités.

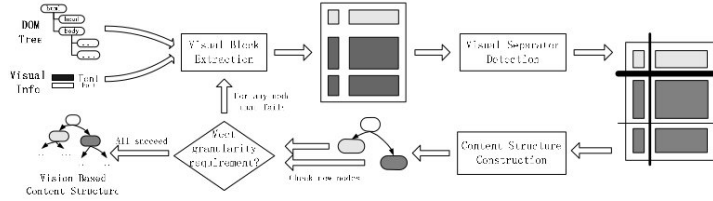


FIGURE 2.4 – Algorithme de segmentation [1]

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.

2.1.5 Synthèse

Méthodes	Globale	Locale	Qualitative	Quantitative	Ascendante	Descendante
Mapping		✓				
Segmentation zone	✓					
Segmentation densitométrique	✓					
Segmentation indice visuel	✓					

2.2 Inférence de structures

2.2.1 Approche fonctionnelle

[2] propose une approche afin de découvrir la fonctionnalité des éléments composant une page web. Les différents éléments 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 la page sous forme d'une structure arborescente dont les feuilles forment des objets basiques et les noeuds ayant des descendants forment des objets plus complexes appelés objets composites. Une fois construite cette structure, les auteurs proposent d'élaborer des fonctions afin de déterminer la sémantique des objets. Les fonctions sont construites par rapport à un modèle classifiant les différents propriétés fonctionnelles des objets et décrivant la relation des objets formant des objets composites. L'hypothèse est que chaque objet au travers de ces fonctionnalités reflète l'intention de son auteur, c'est à dire ça sémantique.

Un objet basique peut, par exemple, apporter une information, déclencher une action de navigation ou simplement rendre plus esthétique la page. Les auteurs proposent les propriétés fonctionnelles décrites ci-dessous :

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

Sémantème : la sémantique associée par le HTML

Décoration : dans quelle mesure l'objet sert à l'amélioration de l'esthétisme de la page

Hyperlien : l'objet possède-t-il des hyperliens et où pointent-ils ? vers un autre objet ? vers une autre page ?

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

Les objets composites sont définis suivant la relation des objets, le composant, entre eux. Ces relations sont classées ainsi :

Relation de Classification :

- Complémentaire : les enfants de la racine sont complémentaire à la réalisation de la fonction
- Parallèle : les enfants de la racine ont une importance égale à la réalisation du but et possède les mêmes propriétés
- Table : Les enfants de la racine peuvent être classifiés vers un enfant racine parallèle

Relation de présentation :

Les objets spécifiques sont des objets propres à un environnement, on leur associe une catégorie. Exemple d'une fonction pour la détection des bars de navigation :

- Les enfants de la racine sont des objets de navigation tel que NN/NR ne soient pas inférieurs à H_{min} , NN est le nombre total d'enfant de la racine navigable, et NR est le nombre total d'enfant racine
- Les enfants de la racine doivent être inférieurs à L_{max}
- Text NNB
- Les enfants de la racine doivent avoir des propriétés similaires

2.2.2 ITF-IDF

Chapitre 3

Langage de publication de page web

3.1 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 <DIV> 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.

3.2 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'explicitier 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

```

<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

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.

3.3 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.)

```

<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

- Document Structure Roles : décrit les structures qui organisent un document (article, définition, entête, ect.)
- 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.

3.4 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 3.4

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).

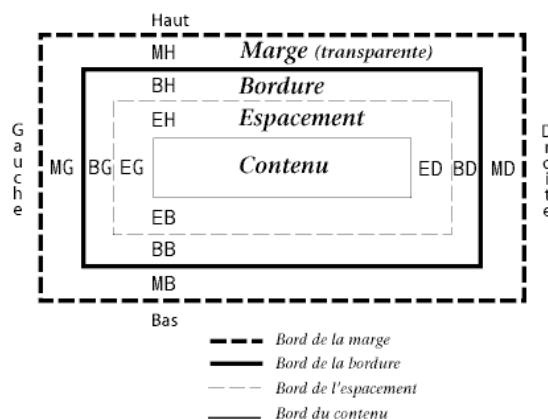


FIGURE 3.6 – Modèle de boîte

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 deuxième 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 exhaustive de propriétés 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 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.*

Chapitre 4

Accessibilité

Une application directe de la personnalisation de page est la possibilité de rendre celle-ci plus accessible à un utilisateur en situation de déficience visuelle. L'élaboration de ce méta-modèle permet de prendre en compte ces aspects. Sans être exhaustive, plusieurs 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 évitées.
- certaines régions de la page ne sont pas utilisées 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 de basse vision, des utilisateurs vont devoir zoomer fortement sur le contenu d'une page. Dans ce contexte, une trop forte différence de 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 5

Réalisation

5.1 Méta-modèle

5.1.1 Introduction

HTML 4 fournit une sémantique, dans ces éléments de langage, pour le décrire des noeuds de contenu (image, texte) mais pas pour des structures logiques plus élaborées (article, menu, bannière etc.). On utilise structure logique dans le sens mathématique du terme, c'est à dire un ensemble muni de fonctions et de relations. Par exemple, la figure 5.1 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 textuelles sont des liens de navigation signalés par la balise `<a>`, 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 préférence sur de telles constructions.

Ce langage fournit une composante syntaxique extrêmement souple. On veut exprimer par composante syntaxique extrêmement souple le fait que HTML 4 autorise la construction d'un ensemble potentiellement infini de structures logiques. D'autre part, le langage ne permet pas d'associer une sémantique explicite aux différentes structures logiques. La sémantique devient de ce fait implicite et est exprimée au travers de la mise en forme des structures. Cette configuration permet ainsi d'associer une même sémantique (implicite) à différentes structures logiques. Si l'on reprend la figure 5.1, on voit que l'on a deux structures logiques différentes ayant la même sémantique (un menu).

La contribution de ce méta-modèle vient apporter une composante 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 3.4) 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. Cela va nous permettre de mieux intégrer l'accessibilité dans la personnalisation des pages web.

```

<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 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>
</div>

```

FIGURE 5.1 – Exemple de différentes conceptions de menu avec HTML4

5.1.2 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. On constate que 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 5.2 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 (<DIV>) ou un paragraphe (<p>) génère un retour à la ligne
- La méta-classe *Inline* dont les instances ne provoquent pas de retour à la ligne. Par exemple, <p>Bonjour tous le monde</p>, la balise ne provoque pas de retour à la ligne dans le paragraphe.

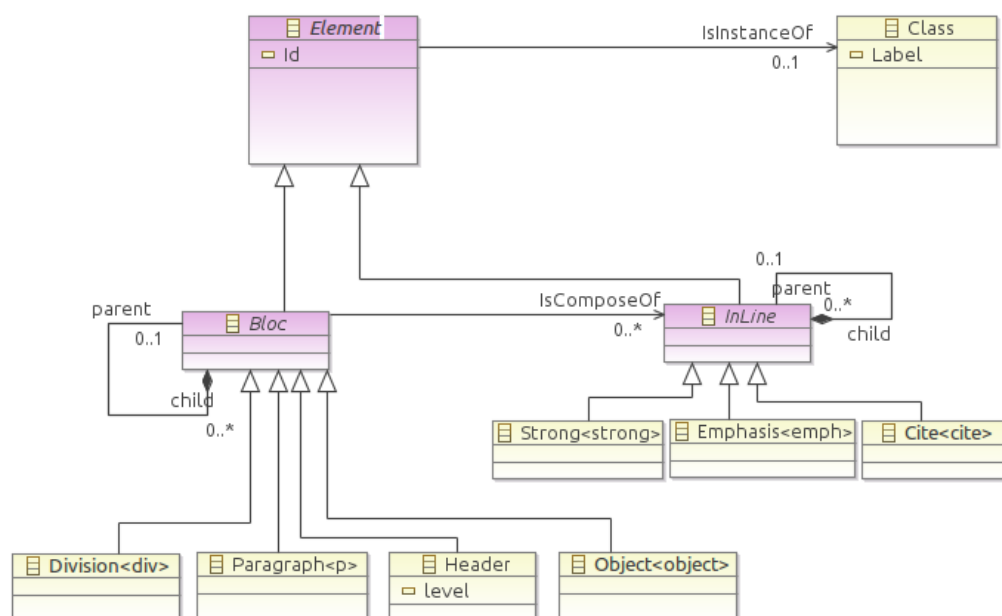


FIGURE 5.2 – Méta-modèle HTML 4

Instance HTML 4 décrit les structures logiques de plus haut niveau à l’aide de d’éléments structurelles générique (*DIV*) et leur propriété de mise en forme (bloc/inline). Si l’on instancie le premier menu de la figure 5.1, 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.

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 5.3 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.

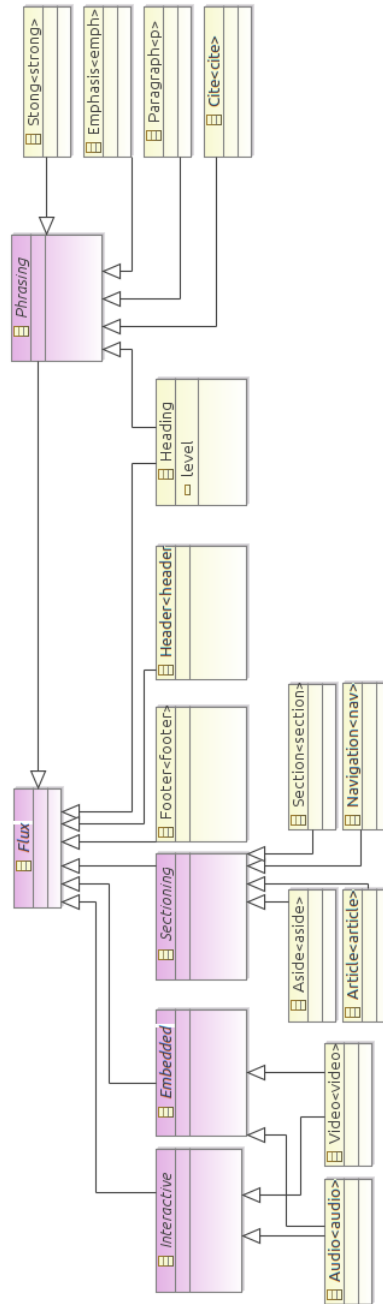


FIGURE 5.3 – Méta-modèle HTML 5

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. 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 5.1, 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.
```

5.1.3 Méta-modèle

5.1.4 Modèle de contenu

La conception de cette partie du méta-modèle repose sur les éléments conceptuels identifiés dans la sous-section 5.1.2 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 5.4 et 5.5) 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 *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 `` déclenche 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.

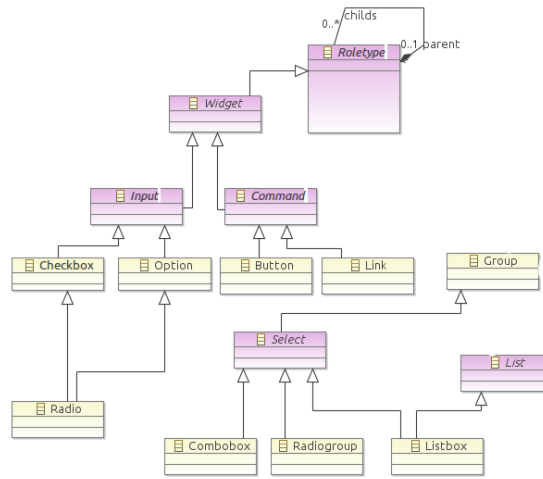


FIGURE 5.4 – Méta-modèle interaction

Modèle de mise en forme

Le méta-modèle de mise en forme (cf. figure 5.6) 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 :

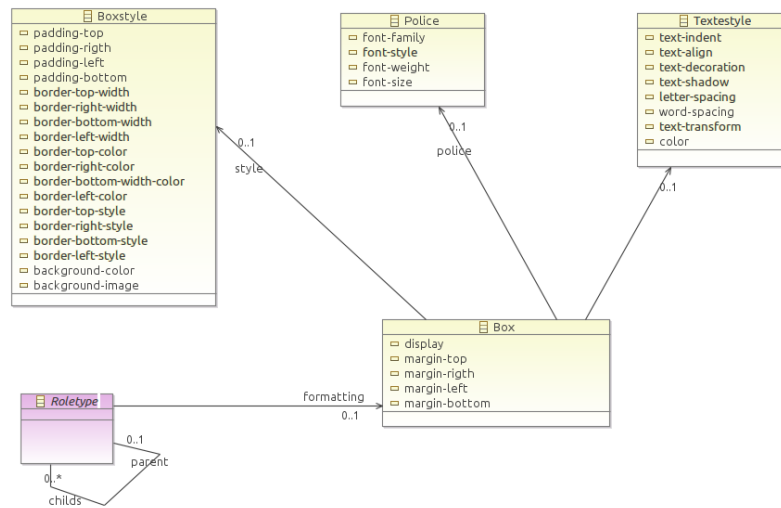


FIGURE 5.6 – Méta-modèle

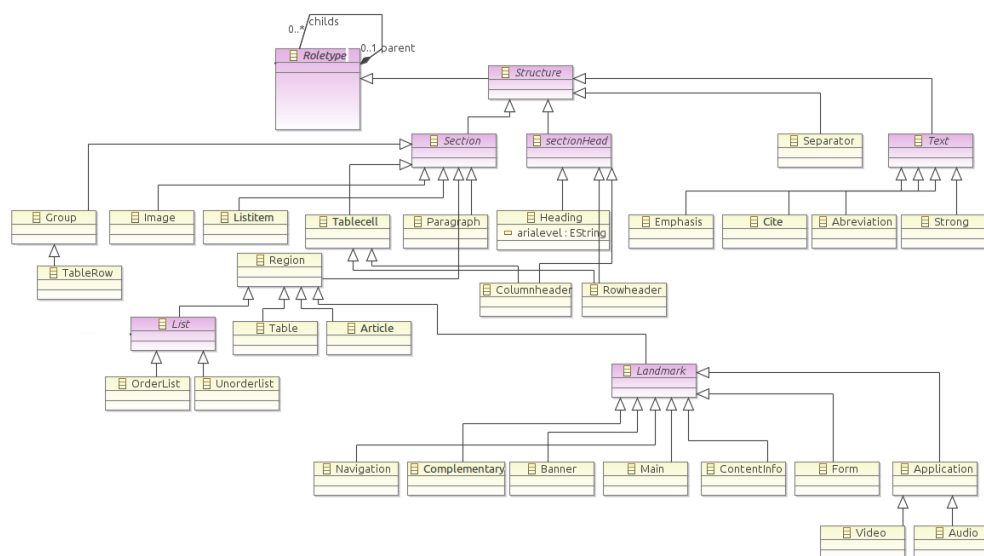


FIGURE 5.5 – Méta-modèle contenu

- 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 *Méta-classe 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

5.2 Extraction de structures

5.2.1 Introduction

Dans cette partie nous traitons de l'extraction des structures logiques dans une page. Nous nous intéressons à la sémantique de ces dernières dans la section 5.3.

Le DOM d'une page web est constituée d'un ensemble de structures logiques. Ces structures logiques sont 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 explicite 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 dans un environnement hétérogène. 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.).

L'extraction de ces structures pose ainsi problème quand le contenu est hétérogène. En effet, dans le cas où le contenu est homogène on peut facilement extraire les différentes structures logiques d'après l'annotation des noeuds. Si l'on regarde le listing 5.1 on a un contenu homogène (un document), les spécifications de HTML4 suffises à l'extraction des structures logiques. L'exemple est un document constitué d'une première division qui est l'introduction, puis une seconde division qui est une section. En revanche, avec un contenu hétérogène on ne peut plus se baser sur les spécifications de HTML4. Si l'on regarde le listing 5.2, la page se compose d'une bannière, d'un document, d'un menu, etc. Du point de vue de notre méta-modèle ce sont des contenus hétérogènes que l'on souhaite extraire. Cela n'est pas possible en se basant uniquement sur la sémantique de HTML4.

Listing 5.1 – 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 5.2 – Exemple contenu hétérogène

```
<div>
<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>
```

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 5.3 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 5.4 on voit que les étiquettes *DIV* ne sont utilisées pour indiquer une division mais pour faire de la mise en forme.

Listing 5.3 – Extrait menu lemonde.fr

```
<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>
```

Listing 5.4 – Pattern page d'accueil google.fr

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

La diversité des structures (*cf.* section 5.1.1) complexifie la solution puisqu'on ne peut pas utiliser de méthodes basées sur la correspondance de structures.

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 la décomposer en structure logique plus fine. La bannière des pages web intègre souvent dans leur contenu des menus et outils de recherche que nous souhaitons extraire.

Toutes ces contraintes posent le cadre de l'élaboration de notre méthode d'extraction des structures logiques dans une page. /*comparer l'existant et expliquer le choix de la méthode*/ /*note segmentation ascendante, d'identifier des sous structure??*/

Pour la conception des différentes heuristiques nous émettons l'hypothèse, comme [1], qu'elles doivent intégrer les propriétés de mise en forme associées aux nœuds du DOM. Notre point de vue est que le lecteur (Humain) va reconnaître les différentes structures qui composent la page principalement les aspects de présentations (typographie, zone de couleur, agencement dans la page, *etc.*). Notre méthode prend en compte ces caractéristiques.

Nous proposons ainsi plusieurs concepts pour l'élaboration d'heuristiques.

5.2.2 Méthode

Description modèle de page

La méthode choisie est une approche basée sur des découpages successives de la structure de la page. Depuis la racine de la structure d'une page, on va parcourir (suivant un parcours infixe) la structure de la page. 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 attribue un degrés de cohérence à la structure. Dans le second cas, on continue le parcours. Si le Degrés de cohérence du bloc est inférieur à un certain seuil, alors on applique récursivement le même processus.

Dans cette partie nous définissons notre vision des briques plus bas niveau de construction d'une page web. L'idée est de définir des concepts détachés des spécifications HTML qui vont nous permettre de prendre la décision de diviser ou non un noeud. En effet, les navigateurs implémentent les spécifications des langages pour au moment de la mise en forme. Cependant Les concepteurs des pages web peuvent tout à fait redéfinir ce comportement au moyen du langage CSS. Nous proposons une classification des nœuds constituant une page web.

Nœuds conteneur : l'ensemble des nœuds qui peuvent contenir d'autres nœuds (conteneur et donnée). Ils ont un rôle de structuration logique. Par exemple, `<P>` peut structurer un ensemble de nœuds de données textuelles pour former un paragraphe.

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 *STRONG* signalent de mettre en gras un texte mais ne structure pas les nœuds qu'elles encapsulent. Si les nœuds conteneurs décrivent une structure logique, les nœuds de mise en forme décrivent une structure physique.

Nœuds de données : l'ensemble des nœuds qui contiennent un contenu atomique. Les contenus atomiques correspondent aux éléments multimédias dans la page : texte, image, video, etc. On inclut les balises ``, `<OBJECT>` et `<H1-H6>`

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

Nœuds description : 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 utilisées pour renseigner le navigateur sur la page

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

Heuristiques

L'élaboration des différentes heuristiques ci-dessous reposent sur les concepts décrits précédemment et intègrent une fonction de distance visuelle qui prend en compte les propriétés associées à chaque nœud telles que la couleur d'arrière plan ou encore les propriétés de bordure des nœuds. Nous définissons également une taille pour chaque nœud, cette taille est calculée suivant la somme du poids de chacun des descendants du nœud puis est pondérée suivant son niveau dans la structure de la page.

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

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

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. Nous lui attribuons un degrés de cohérence de 10.

règle 4 : Si l'un des enfants du nœud est un nœud de conteneur alors on parcourt ce nœud

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 parcourt le nœuds

règle 7 : Si le premier enfant valide du noeud est un noeud de titre alors on alors on extrait ce nœuds comme une structure logique. Le Degrès de cohérence est calculé suivant le type de noeud pondéré par l'importance du noeud.

règle 8 : Si la fonction de distance visuelle est vérifiée (renvoie vraie) pour au moins un enfant du nœuds alors on parcourt 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 2 La règle 2 permet de découvrir des structures logiques plus intéressantes dans une page. D'après l'exemple 5.5, cette règle va permettre d'atteindre le noeud *center* contenant la page la bannière et la bar de recherche. Les balises *DIV* sont détourné de leur sémantique de division de contenu pour être utilisé pour appliquer différents styles sur un élément 5.4. La règle 2 permet de parcourir l'arbre de la page jusqu'à la structure logique.

Listing 5.5 – 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. La règle donne le degrés de cohérence maximum, 10.

Listing 5.6 – 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 5.5 après l'application de la règle 2 on va découvrir des structure logique plus intéressante qui sont le bannière de la page d' accueil de GOOGLE et la bar de recherche.

Règle 5

Règle 6 Dans 5.7 le noeud *mw-navigation* n'est pas un noeud

Listing 5.7 – Pattern wikipedia.fr

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

Règle 7

Règle 8

Règle 9

5.3 Inférence de structures

Chapitre 6

Validation

Chapitre 7

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 Élément 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
```

☐ I have a bike
☐ 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"/>

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

☒ Male
☐ 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)

Caractéristique	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
```

☒ Rouge
☐ Blanc
☐ 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>
```

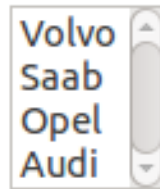


FIGURE 4 – Exemple de Select

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

Caractéristique	Valeur
Abstrait	Non
Super Classe	Select
Sous Classe	
Alignement HTML	<i>cf. figure 5</i>

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).

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

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

Caractéristique	Valeur
Abstrait	Non
Super Classe	List
Sous Classe	
Alignement HTML	

OL(order list) Liste ordonnée d'éléments

Caractéristique	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 (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 Élément 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*.

Caractéristique	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.

Caractéristique	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.

Caractéristique	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 ??).

Caractéristique	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 ??).

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 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 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 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 conçu 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 permettent 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] : arrière-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 [?]. 15

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.