Ministère de l'enseignement supérieur
et de la recherche scientifique
Université de M'sila
Faculté des Mathématiques et d'Informatique
Département d'Informatique

Mémoire de fin d'études
Pour l'obtention de Diplôme **d'ingénieur**
D'état en informatique

Spécialité : Informatique
Option: Systèmes d'Information Avancées (SIA)

Thème

Développement d'un Système D'extraction des données à partir des sites web

Réalisé par :
Zerroug Abdelraouf

Dirigé par :
Mr. Mehenni Taher

2010 /2011

Remerciements

Je tiens à remercier Monsieur Mehenni Tahar , pour l'intérêt qu'il m'a porté pendant la durée du projet, pour ses remarques constructives, son aide lors du suivi.

Je remercie sincèrement Monsieur Dhif Rabie, pour son aide précieux.

Je remercie également toutes les personnes qui ont contribué de près ou de loin à la réussite de ce projet, sans oublier mon entourage qui ont supporté toutes ces longues années d'études.

Dédicaces

Ahmadou Allah, d'abord, de m'avoir donné les moyens d'arriver jusqu'a là.

Je dédie ce travail :

A l'hommage de ma mère

A mon père , qui est toujours derrière moi à tout moment.

A ma chère épouse et mes chèrs enfants (Leila ,Mohamed Fawzi , Taha Abdelbary)

A mes frères et mes sœurs ;

A tous les membres de ma grande famille ;

A tous mes amis ;

A tous ceux qui me sont chers ;

Zerroug Abdelraouf

Sommaire

Liste des figures et des tableaux

Introduction

Jour après jour, l'internet devient la source la plus importante d'informations dans notre vie. Les sites web constituent de nos jours les moyens les plus utilisés des technologies web.

Néanmoins, la restructuration des sites web et l'éventuelle rétro-conception de leurs éléments constitutifs peuvent s'avérer plus qu'intéressantes. Nous pouvons énumérer certaines situations où le recours au retour arrière du cycle de vie des sites web est obligatoire.

- Lorsqu'on est en présence d'un site web statique ; et qu'au fur et à mesure le site web devient important et volumineux, on peut penser à sa transformation en un site web dynamique et donc à la construction d'une base de données à partir de ses pages html.
- Lorsque l'administrateur du site web dynamique abandonne sa mission d'administration du site (pour une raison ou une autre), le propriétaire du site est obligé de faire appel à un autre administrateur pour la prise en charge du site. Celui-ci devra certainement reconstruire la base de données du site à partir de ses pages html.

Afin d'améliorer les possibilités d'exploitation, il est donc intéressant d'élaborer des méthodes pour séparer les données de leur présentation, les extraire, les interpréter et retrouver leur structure sémantique invisible dans les pages *HTML*.

Pour cela nous avons pensé à l'élaboration d'un system d'extraction de données à partir de page Html du site dans le but de reconstituer la base de données. Les données et structures extraites deviennent une base pour des applications aussi diverses que la réingénierie de site.

Chapitre I

CONCEPTS FONDAMENTAUX DE LA RETRO-INGENIERIE

Chapitre I
CONCEPTS FONDAMENTAUX DE LA RETRO-INGENIERIE

1. Introduction

Le présent chapitre est une introduction générale et un aperçu sur les concepts de rétro-ingénierie. Avant d'entreprendre les définitions et aspects de la rétro-ingénierie, il est nécessaire de localiser cette dernière dans son contexte. Il faut noter que la rétro-ingénierie est une activité liée étroitement à la maintenance des systèmes. Nous présentons dans le reste du chapitre quelques aspects de la maintenance avant de nous intéresser à la rétro-ingénierie.

2. La maintenance

2.1. Définition de la maintenance

La maintenance n'est pas seulement la détection et la correction des erreurs trouvées dans un programme. On trouve différentes définitions de la maintenance logicielle. Des acteurs important du domaine définissent l'activité de la façon suivante :

1. L'organisme de standardisation des logiciels, L'IEEE, définie dans la référence « software maintenance standard », IEEE STD 1219-1993, le terme de maintenance logiciel comme ceci : « la modification d'un logiciel après son entré en production, afin de corriger ses erreurs, d'améliorer ses performances et autres attributs, ou pour adapter le produit à son environnement »

2. Martin et MC Clure, éminents théoriciens du domaine définissent la maintenance comme « Les changements effectués sur les programmes informatiques après livraison à leurs utilisateurs »

3. Von Mayrhauser: « La maintenance couvre le cycle de vie des systèmes logiciels à partir de leur mise en production jusqu'à la fin de leur utilisation. »

Le thème commun de ces multiples définitions est que la maintenance est une activité qui poursuit la réalisation d'un travail. L'activité de maintenance commence dès la mise en production du logiciel [5].

2.2. Pourquoi la maintenance est-elle nécessaire ?

Pour répondre à cette question, on se doit simplement d'observer la mise en production d'un logiciel livré à des utilisateurs. Lorsque les premiers utilisateurs vont utiliser les produits, ils vont trouver de nombreux problèmes, et disfonctionnements, et trouver d'innombrables fonctions à rajouter. Ces changements vont être demandés aux mainteneurs de l'application, qui vont corriger et améliorer le système. Puis les utilisateurs vont s'adapter aux changements et nouvelles fonctionnalités et à nouveau perturber les mainteneurs. Dans la plupart des cas, le cycle de maintenance d'un logiciel constitue la plus grosse part de son cycle de vie, en temps et en argent. Cette maintenance sans fin se justifie par le fait qu'une application doit sans cesse évoluée pour continuer à être utilisée, et ne pas mourir.

2.3. Quatre types de maintenance

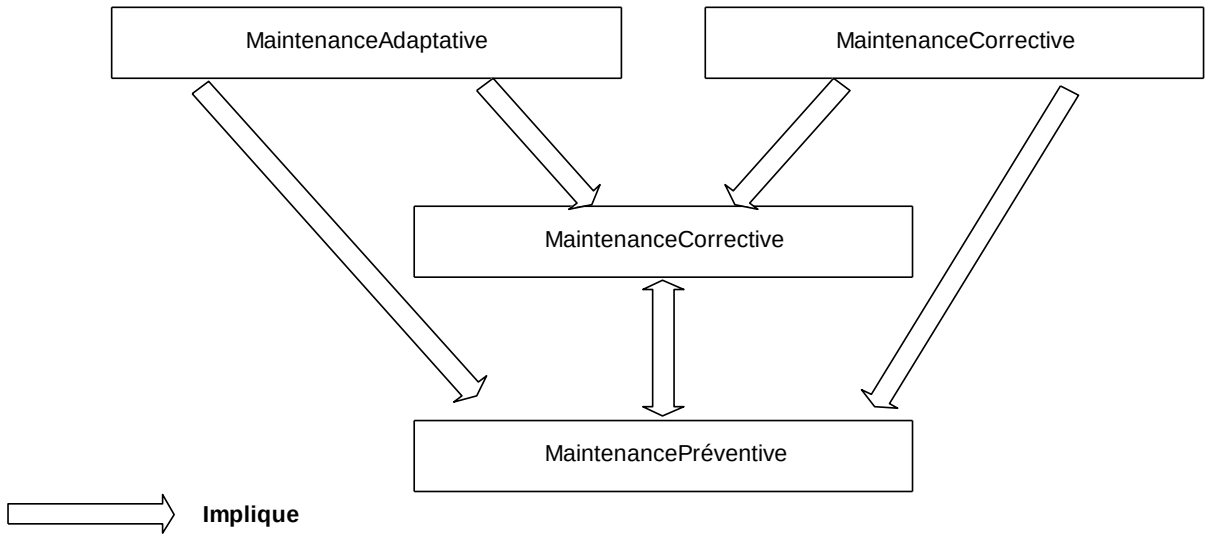


Fig. 1.1 Relations potentielles entre les différents types de maintenance logicielle [5]

2.3.1. A. Maintenance corrective :

La maintenance corrective ce réfère aux modifications initiées dûe aux défauts du logiciel. Ces défauts peuvent résulter d'erreur de spécifications, de choix d'architecture ou d'erreur de programmation, ou de tests invalides ou incomplets. Les défauts peuvent également provenir de problèmes de performances ou de traitements de données. Ces erreurs sont souvent nommées «erreurs résiduelles », empêchant le logiciel d'être

conforme aux spécifications. Lors de l'apparition de problèmes bloquants sur un système, la maintenance résout le problème en proposant des « patchs ».

2.3.2. B. Maintenance perfective :

Ce terme est utilisé afin de décrire les changements mis en œuvre afin d'étendre le système et d'améliorer l'efficacité d'un système d'information en exploitation. Une application tend à devenir performante grâce à la succession de changements résultant de l'accroissement des fonctionnalités. Imaginons une application faite sur l'architecture constituée d'un noyau central et des multiples modules implémentant les fonctionnalités du logiciel. On peut dans ce cas, ajouter à volonté de nouvelles fonctionnalités, à la demande des utilisateurs. Durant les diverses étapes, le système évolue grâce à l'ajout de petit programme, ou modules, facilement intégrable, aisément maintenable. On constitue ainsi une grosse application offrant une bonne résistance à la modification. Ce peut donc être l'ajout de nouvelles fonctionnalités ou la redéfinition de fonctions existantes, ou la suppression, de fonctions inutiles ou redondantes, alourdissant inutilement le système.

2.3.3. C. Maintenance adaptative :

Elle a pour but d'effectuer les modifications nécessaires pour adapter un système d'information aux changements. Même en l'absence d'erreur résiduelle, une application est vouée à changer pour s'adapter à l'environnement. Lorsque l'on emploie le terme d'environnement dans notre contexte, on se réfère aux conditions influençant le système : Par exemple, les règles fiscales, les changements de stratégies commerciales, les changements de législation, les changements de plates-formes, ou de matériel. Tous ces facteurs sont des éléments déterminant l'adaptation des logiciels, à de nouvelles contraintes.

La maintenance adaptative inclus tout travail autour d'une plate-forme, d'un OS ou d'un compilateur différent. Elle peut être rendue nécessaire afin d'améliorer les performances d'une application pour cause de demandes accrues et donc migrer par exemple d'un système séquentiel de traitement de taches, vers un système de traitement en parallèle. Similairement, des applications peuvent être portées sur d'autres plates-formes afin de tirer avantage d'un nouvel OS, ou le support d'une nouvelle base de données, pour des gains de performances, une sauvegarde plus facile, ou une résistance à la défaillance accrue. Un exemple de maintenance adaptative, pour cause de changement de règle économique, est le passage à l 'Euro. Tous les systèmes d'informations bancaires ont dû être adaptés pour supporter la nouvelle monnaie [5].

2.3.4. D. Maintenance préventive :

La maintenance préventive a pour but d'anticiper tous les problèmes possibles. Les phases de maintenance corrective, adaptative, et perfective peuvent être considérées comme des travaux qui détériorent généralement peu à peu la structure de l'application si aucun travail de maintien n'est réalisé. Ce travail constitue des changements préventifs. La maintenance préventive est une entreprise de prévention de disfonctionnement, ou d'amélioration de la maintenabilité du logiciel.

Ces changements sont souvent initiés par le département de maintenance souhaitant rendre plus simple la compréhension ou la structure du programme pour des futurs travaux de développement. Les changements préventifs ne donnent généralement pas d'amélioration substantielle des fonctions de base. Le changement préventif peut être une restructuration du code, ou la mise à jour de la documentation. La mise à jour du système documentaire est fréquemment oubliée. Les documents affectés par les changements doivent être modifiés afin de refléter l'état courant du système.

En principe, les activités de maintenance logicielle peuvent être classées dans ces quatre catégories. En pratique, elles s'entremêlent souvent. Par exemple, lors de l'adaptation d'une application à un nouvel OS, donc une activité d'adaptation, de curieux bugs peuvent apparaître. On effectue donc une correction dans une maintenance corrective. Similairement, l'activité d'amélioration d'algorithme peu impliquer la restructuration du code, et donc une activité de maintenance préventive. Pourtant, malgré l'enchevêtrement de ces diverses activités, il est bon de garder la distinction entre les différents types de changements. En premier lieu, ceci permet d'allouer des niveaux de priorités aux demandes de changements. Certains changements requièrent des réponses plus rapides que d'autres.

3. La rétro-ingénierie

La rétro-ingénierie aussi appelée reverse engineering a pour but de remonter les phases de conception d'un système. Dans le logiciel c'est le moyen de partir du code et de remonter vers la modélisation, voir la spécification on nomme cela la rétro-conception.

La rétro-ingénierie est très utile pour comprendre la structure d'un programme peu documenté, voir absolument pas documenté. Le diagramme de classe obtenu par rétro-conception peut nous permettre de comprendre globalement le logiciel, les dépendances entres classes et packages. Cette méthode permet également à gérer l'interopérabilité entre logiciels en réussissant à décompiler les protocoles de transfert ou les cryptages dans certains cas.

Après des premières années caractérisées par une terminologie foisonnante, les bases conceptuelles de la rétro-ingénierie et le vocable associé ont été établis en 1990 par Chikofsky et Cross dans l'article "Reverse Engineering and Design Recovery: a Taxonomy" (voir table 1.1). Cette taxonomie fait toujours référence et c'est donc sur ces définitions qu'est basé cet article. Comme le montre la Figure 1.2, extraite de la taxonomie est basée sur une classification des transformations qu'il est possible d'appliquer au logiciel. Le concept de transformation est ainsi au cœur de la rétro-ingénierie.

La légende originelle de la Figure 1.1 indique : "la rétro-ingénierie et les processus associés sont des transformations inter ou intra-niveaux d'abstraction". Quelques années plus tard, Arnold raffine cette taxonomie en fournissant "une procédure décisionnelle pour classifier les transformations du logiciel" .

Ingénierie directe *Forward engineering*	Forward engineering is the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system
Rétro-ingénierie *Reverse engineering*	Reverse engineering is the process of analyzing a subject system with two goals in mind: (1) to identify the system's components and their interrelationships; and, (2) to create representations of the system in another form or at a higher level of abstraction
Redocumentation	Redocumentation is the creation or revision of alternate views semantically coherent with the examined System
Rétro-conception *Design recovery*	Design recovery is a subset of reverse engineering in which domain knowledge, external information, and deduction or fuzzy reasoning are added to the observations of the subject system to identify meaningful higher level abstractions beyond those obtained directly by examining the system itself
Restructuration *Restructuring*	Restructuring is the transformation from one representation to another at the same relative abstraction level, while preserving the subject system's external behaviour
Re-ingénierie *Reengineering*	Reengineering is the examination of a subject system to reconstitute it in a new form and the subsequent implementation of the new form

Tab. 1.1 Taxonomie de Chikofsky and Cross [6].

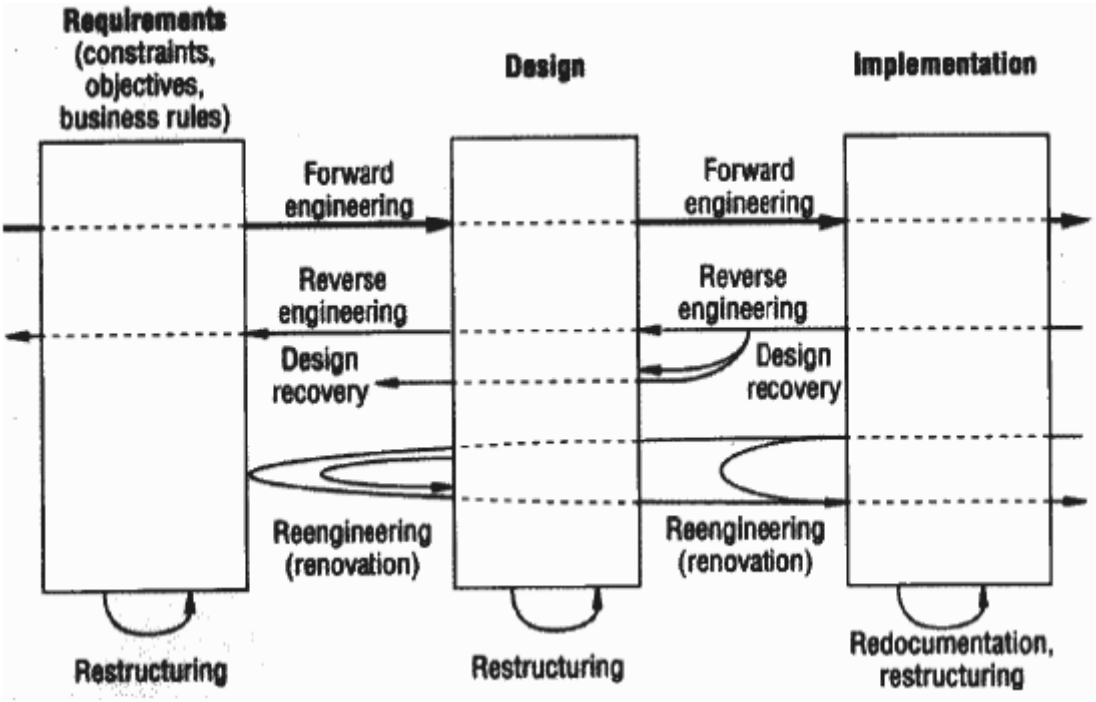


Fig. 1.2 Classification des transformations [6].

3.1. Cycle de vie d'un projet de rétro-ingénierie

La figure 1.3 donne le cycle de vie du projet de rétro-ingénierie. Il est constitué de trois phases :

1- La phase de reconstitution de l'architecture : qui consiste à l'élaboration à partir du système hérité, de nouveaux concepts conduisant à mieux à comprendre le fonctionnement du système.

2- La phase de transformation de l'architecture : elle consiste à la mise en œuvre des outils nécessaires à la transformation de la nouvelle architecture du système.

3- La phase de développement basé sur la nouvelle architecture : elle consiste à entreprendre le développement du nouveau système basé sur la nouvelle architecture.

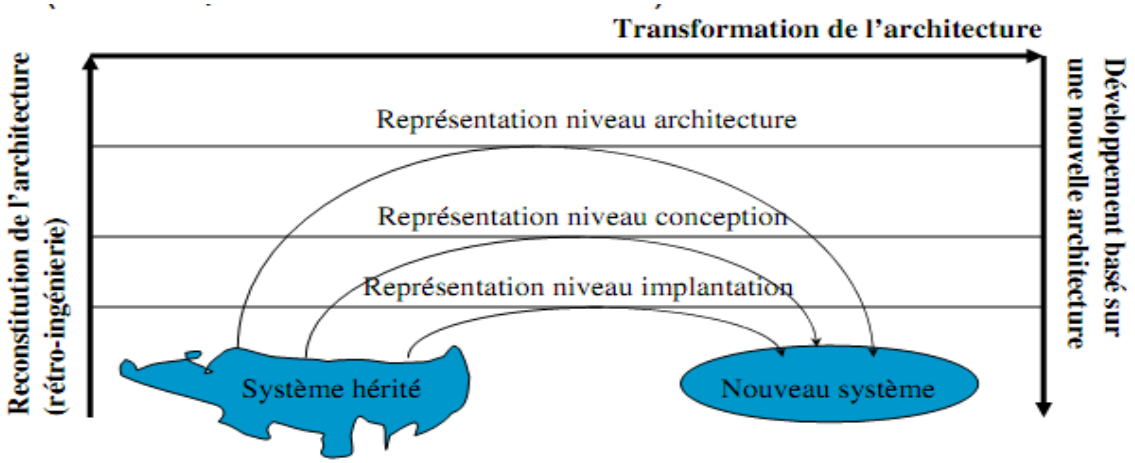


Fig 1.3 Cycle de vie d'un projet de rétro-ingénierie[1]

3.2. Architecture des outils de rétro-ingénierie

Au cours des années 90, de très nombreux outils de rétro-ingénierie ont été proposés. Arnold a fait remarquer que tous ces outils partageaient globalement l'architecture présentée dans la Figure 1.4. A gauche on trouve les artefacts logiciels à partir desquels sont extraites les informations. Les vues produites à droite peuvent être présentées aux ingénieurs sous de multiples formes.

Au point de vue des techniques classiques de rétro-ingénierie on peut bien évidemment citer les techniques d'analyse du logiciel (partie gauche du diagramme), mais aussi les techniques de visualisation (partie droite) qui sont alors particulièrement adaptées et l'on souhaite typiquement obtenir des "cartographies" qui sont des modèles visuels des applications logicielles existantes. Dans le monde de la rétro-ingénierie, on parle souvent d'environnement d'exploration des logiciels, car il s'agit de "naviguer" d'un modèle à l'autre et ce de manière interactive.

L'un des défis auquel les environnements de rétro-ingénierie doivent faire face consiste à prendre en compte la multiplicité des sources d'informations disponibles. Cet aspect n'a hélas pas été clairement identifié dans les travaux fondateurs en rétro-ingénierie. Par exemple remarquons qu'une et une seule boite est présente à gauche de la Figure 1.4. Bien que cette boite soit à juste titre appelée "produit logiciel", elle a été pendant très longtemps assimilée à tort au code source, qui n'est en fait qu'un aspect particulier du logiciel.

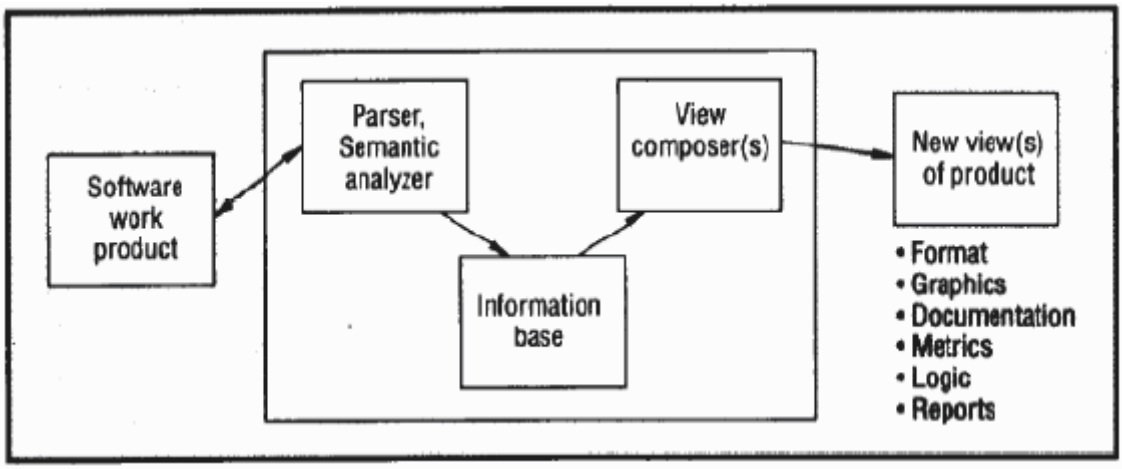


Fig. 1.4 Architecture des outils de rétro- ingénierie[7]

3.3 Caractéristiques des outils de rétro-ingénierie

La première génération d'outils de rétro-ingénierie correspondait à des outils construits de manière monolithique et "câblée" centrée sur une fonctionnalité précise. Ces outils étaient typiquement liés à un langage de programmation unique, proposaient un seul type d'analyse et un seul type de vue. C'est le cas par exemple des outils permettant à partir de programmes C d'extraire un graphe d'appels entre fonctions et d'afficher cela sous forme graphique. Ces solutions ad hoc, câblées et monofonctionnelles ne sont plus suffisantes dans le contexte Rétro-ingénierie a'aujourd'hui. On assiste à une multiplication (1) des sources de données en entrée, mais aussi (2) des types de vues à produire en sortie. Chacun de ces deux points est étudié ci-dessous.

La problématique liée à la multiplicité des sources de données en entrée dépasse largement l'aspect multi-langages au sens classique, même si celui-ci a été catalogué comme l'un des points difficiles mais essentiel à résoudre en pratique. Désormais il s'agit en effet de prendre en compte une très large gamme d'artefacts logiciels qui inclut par exemple code source, bytecode, binaires, librairies, traces, fichiers de configurations, rapports de bug, etc.

Les environnements de rétro-ingénierie doivent pouvoir extraire des informations provenant de sources d'informations très variées.

Les environnements de rétro-ingénierie doivent pouvoir générer un ensemble de vues variées et adaptées à chaque type d'intervenant [2].

La plupart des outils classiques de rétro-ingénierie disponibles étant "câblés", ils n'offrent que très peu de souplesse. Sachant qu'ils ne couvrent généralement que certaines

fonctionnalités, l'une des difficultés récurrentes des projets de rétro-ingénierie consiste à trouver une manière de couvrir tous les besoins du projet en intégrant autant que faire se peut les outils dont on dispose. Ainsi l'interopérabilité des outils de rétro-ingénierie est devenue un thème de recherche important. L'une des approches les plus courantes consiste à faire communiquer les différents outils via l'import et l'export de données sous la forme de fichiers écrits dans des standards adaptés à la représentation de graphes, tel GXL. Cette approche offre bien peu de souplesse. Non seulement il est nécessaire d'écrire des "wrappers" permettant de faire la correspondance entre les structures internes manipulées par les outils et les formats externes, mais en plus ces solutions sont basées sur l'échange permanent de gros volumes de données entre outils (typiquement via des fichiers XML). Une telle solution ne permet pas non plus l'intégration fine des outils dans les environnements de programmation existants. L'ajout de nouvelles fonctionnalités est également difficile car il est nécessaire de posséder de nombreuses expertises techniques : analyse syntaxique, transformations vers des formats d'échanges, interface homme-machine, visualisation du logiciel, etc. Toutes ces difficultés résultent dans des processus de rétro-ingénierie lourds et peu flexibles car entièrement contraints par les outils existants.

En fait, de part la nature exploratoire des processus de rétro-ingénierie, l'un des défis qu'il est nécessaire de surmonter est de minimiser ces coûts de développement de sorte que l'on puisse adapter les outils "à la demande" au cours du déroulement de chaque projet de rétro-ingénierie.

L'un des défis à surmonter est de pouvoir passer de processus de rétro-ingénierie rigides et figés à des processus de rétro-ingénierie agile, c'est-à-dire dans laquelle la flexibilité est essentielle et où les outils sont développés à la demande au fur et à mesure du processus.

4.Conclusion

Ce chapitre a survolé les princiaux concepts liés à la rétro-ingénierie. Nous décrirons dans le chapitre suivant un autre volet très lié à notre projet, à savoir les sites web ainsi que les langages html et XML.

Chapitre II

NOTIONS GENERALES DU WEB

Chapitre II
NOTIONS GENERALES DU WEB

1. Introduction

On appelle «**Web**» (nom anglais signifiant «**toile**»), contraction de «*World Wide Web*» (d'où l'acronyme *www*), une des possibilités offertes par le réseau Internet de naviguer entre des documents reliés par des liens hypertextes.

Le concept du Web a été mis au point au CERN (Centre Européen de Recherche Nucléaire) en 1991 par une équipe de chercheurs à laquelle appartenaient Tim-Berners LEE, le créateur du concept d'hyperlien, considéré aujourd'hui comme le père fondateur du Web.

Le principe de web repose sur l'utilisation d'hyperliens pour naviguer entre des documents (appelés «**pages web**») grâce à un logiciel appelé **navigateur** (parfois également appelé *fureteur* ou *butineur* ou en anglais *browser*). Une page web est ainsi un simple fichier texte écrit dans un langage de description (appelé HTML), permettant de décrire la mise en page du document et d'inclure des éléments graphiques ou bien des liens vers d'autres documents à l'aide de balises [10].

Au-delà des liens reliant des documents formatés, le web prend tout son sens avec le protocole HTTP permettant de lier des documents hébergés par des ordinateurs distants (appelés serveurs web, par opposition au client que représente le navigateur). Sur Internet les documents sont ainsi repérés par une adresse unique, appelée URL, permettant de localiser une ressource sur n'importe quel serveur du réseau internet.

2. Qu'est-ce qu'un site web ?

Un site web (aussi appelé site internet ou page perso dans le cas d'un site internet à but personnel) est un ensemble de fichiers HTML stockés sur un ordinateur connecté en permanence à internet et hébergeant les pages web (serveur web).

Un site web est habituellement architecturé autour d'une page centrale, appelée «page d'accueil» et proposant des liens vers un ensemble d'autres pages hébergées sur le même serveur, et parfois des liens dits «externes», c'est-à-dire de pages hébergées par un autre serveur.

Une URL se présente sous la forme suivante : http://www.commentcamarche.com/www/www-intro.php3

2.1 Qu'est-ce qu'un site statique?

On entend par page statique, non pas une page sans mouvements ou sans animations, mais une page visible telle qu'elle a été conçue.
Ces pages peuvent présenter toute forme de contenu, animations flash, images, musique, vidéo etc... mais elles sont toujours présentées de la même façon. Elles ne changent pas et c'est en ce sens qu'elles sont statiques.

2.2 Qu'est-ce qu'un site dynamique?

En opposition aux pages statiques, les pages dynamiques permettent de présenter les informations de différentes manières selon l'interaction avec le visiteur.
Les pages sont alors construites "à la volée" grâce à une programmation conçue par le webmaster. Le contenu est issu d'une base de données en fonction de critères établis par l'internaute puis mis en page en temps réel.

C'est le cas par exemple d'un site d'E-commerce: présentation des articles par thèmes, couleurs, prix etc...

C'est également le cas des blogs et des forums où les visiteurs peuvent participer au contenu du site. C'est aussi le cas d'un système de mises à jour[10].

3. Langage HTML

Le **HTML** (« *HyperText Mark-Up Language* ») est un langage dit de « marquage » (de « structuration » ou de « balisage ») dont le rôle est de formaliser l'écriture d'un document avec des balises de formatage. Les balises permettent d'indiquer la façon dont doit être présenté le document et les liens qu'il établit avec d'autres documents.

Le langage HTML permet notamment la lecture de documents sur Internet à partir de machines différentes, grâce au protocole HTTP, permettant d'accèder via le réseau à des documents repérés par une adresse unique, appelée URL.

On appelle **World Wide Web** (noté *WWW*) ou tout simplement **Web** (mot anglais signifiant *toile*) la "toile virtuelle" formée par les différents documents (appelés « **pages web** ») liés entre-eux par des hyperliens.

Les pages web sont généralement **organisées** autour d'une page d'accueil, jouant un point central dans la navigation à l'aide des liens hypertextes. Cet ensemble cohérent de pages web liées par des liens hypertextes et articulées autour d'une page d'accueil commune est appelée **site web**.

Le Web est ainsi une énorme archive vivante composée d'une myriade de sites web proposant des pages web pouvant contenir du texte mis en forme, des images, des sons, des vidéo, etc.

3.1 Le HTML est un standard, mais…

Il est important de comprendre que le langage HTML est un standard, c'est-à-dire qu'il s'agit de recommandations publiées par un consortium international : le World Wide Web Consortium (**W3C**).

Les spécifications officielles du HTML décrivent donc les "instructions" HTML mais en aucun cas leur implémentation, c'est-à-dire leur traduction en programmes d'ordinateur, afin de permettre la consultation de pages web indépendamment du système d'exploitation ou de l'architecture de l'ordinateur.

Toutefois, aussi étoffées les spécifications soient-elles, il existe toujours une marge d'interprétation de la part des navigateurs, ce qui explique qu'une même page web puisse s'afficher différemment d'un navigateur Internet à l'autre.

De plus, il arrive parfois que certains éditeurs de logiciels ajoutent des instructions HTML propriétaires, c'est-à-dire ne faisant pas partie des spécifications du W3C. Ainsi des pages web contenant ce type d'instruction pourront être parfaitement affichées sur un navigateur et seront totalement ou en partie illisibles sur les autres, d'où la nécessité de créer des pages web respectant les recommandations du W3C afin de permettre leur consultation par le plus grand nombre [10].

4. Langage XHTML

Depuis le 26 janvier 2000, le XHTML est la nouvelle norme du W3C en matière de langage balisé pour concevoir des documents Web. Que vos pages existantes soient actuellement conformes ou non aux différentes versions du HTML importe peu. Vous allez rapidement constater que les convertir en XHTML n'est pas sorcier du tout. En effet, puisque le XHTML n'est rien de plus que du HTML reformulé de façon à respecter les règles strictes du XML, il ne vous suffit que d'apprendre quelques règles syntaxiques propres à XML pour commencer à coder selon les normes du W3C. Si vos documents sont déjà conformes (valides) aux règles du HTML 4.01, votre travail de conversion en

sera grandement facilité. Cependant, si vous avez l'habitude de coder selon les règles plus permissives du HTML des premières versions, truffé de balises propriétaires ou dépréciées depuis, vous aurez un peu plus de pain sur la planche.

Tout ce qui vous sépare de votre but, c'est un peu moins d'une dizaine de petites lois et quelques principes d'application. Issues de la spécification XML, ces lois permettent une séparation logique entre les aspects de structure et de présentation dans un document Web. Car voilà réellement ce qu'est le XHTML : un pont entre le HTML (le langage d'hier) et le XML (le langage de demain) [11].

Chaque balise nécessite une fermeture

Dans les premières heures du HTML, on pouvait se permettre d'être relativement brouillon dans la façon d'organiser son code. Maintenant, selon les règles plus strictes du XML, il n'y a plus de place pour une telle permissivité ; ainsi, toutes les balises présentes dans un document Web doivent dorénavant être correctement fermées : il ne faut jamais oublier d'ajouter la balise de fermeture d'un élément quand celle-ci existe :

Invalide :

Lorem ipsum dolor sit amet. Praesent vel justo.

Valide :

Lorem ipsum dolor sit amet. Praesent vel justo.

Même celles qui n'en ont pas

En revanche, comment peut-on fermer ces autres éléments ne possèdant pas de balise de fin, comme les br et img ? En leur inventant une balise de fermeture ? Eh bien presque. Sauf que vous n'aurez pas à les inventer puisque le W3C s'en est déjà chargé pour vous. Vous pourriez effectivement vous mettre à coder des
</br> ou des <img></img>, mais une telle pratique est déconseillée puisqu'il est possible que cela produise des résultats inattendus dans certains butineurs. Ce serait aussi un peu inutile puisque selon la syntaxe XML, il est possible de simplement fermer un élément en lui attribuant une barre oblique (un slash) en fin de balise, comme ceci :
, ou encore <img />. Cependant, si vous optez pour la seconde méthode, il ne faut pas oublier d'inclure un espace entre le contenu de l'élément et la barre oblique, car autrement, les anciens navigateurs, en particulier Netscape 4.x, ne pourront l'interpréter et l'ignoreront tout bonnement :

Incorrect :

Correct :

Imbriquer correctement les éléments

Quand on ouvre une série de balises en cascades, (les unes à l'intérieur de l'espace de définition des autres), il faut obligatoirement les refermer dans l'ordre inverse de l'ordre d'ouverture pour respecter la structure logique interne du document. Il faut toujours voir une balise HTML comme étant incluse dans une autre balise qui lui sert de parent. Ainsi, dans l'exemple ci-dessous, l'élément strong, qui est un enfant direct de l'élément p, doit impérativement se refermer à l'intérieur de l'élément qui le contient :

Invalide :

Paragraphe avec texte en gras

Valide :

Paragraphe avec texte en gras

Utiliser des minuscules dans les balises et leurs attributs

En HTML, on pouvait à loisir utiliser des majuscules ou des minuscules pour baliser nos documents. Certains voyaient même en l'utilisation des majuscules un moyen efficace pour repérer le code HTML du contenu dans un document. Avec XHTML, ce n'est plus possible ; puisque XML est sensible à la casse, toutes les balises et tous leurs attributs doivent obligatoirement être écrits en lettres minuscules. C'est donc dire que les balises LI et li ne sont plus identiques en XHTML, alors que c'était le cas en HTML. Les valeurs d'attributs, par contre, peuvent toujours être écrites en majuscules :

Invalide :

<TEXTAREA ID="monTexte"></TEXTAREA>

Valide :

```
<textarea id="monTexte"></textarea>
```

Chaque valeur d'attribut doit être entre guillemets

Avec les versions antécédentes de HTML, le recours aux guillemets pour encadrer les valeurs d'attributs était conseillé, mais pas obligatoire. Toujours selon les règles de XML, l'utilisation des guillemets n'est plus une proposition, mais bien une obligation. De plus, il ne peut plus y avoir de saut de ligne dans la définition d'une valeur donnée. Les caractères d'espacement (comme les espaces, les bris de lignes et les retours de chariots) sont interprétés différemment de navigateurs en navigateurs. Lorsque des caractères d'espacement sont insérés dans les valeurs d'attributs, les navigateurs tronquent ces espaces et les transposent en code ASCII, d'où parfois d'imprévisibles maux de têtes. Afin de vous éviter ces tracas, prenez simplement la bonne habitude de ne pas laisser d'espace dans vos valeurs d'attributs :

Invalide :

```
<div id=monMenu></div>
```

Valide :

```
<div id="monMenu"></div>
```

Les formes abrégées d'attributs sont interdites

Certaines balises en HTML possédaient des attributs autonomes qui pouvaient être utilisés sans valeurs associées, comme c'était par exemple le cas pour la balise input avec laquelle on pouvait utiliser les attributs "checked", "disabled" et "readonly". Une pratique courante en HTML consistait donc à les déclarer de manière abrégée, afin d'économiser le nombre de caractères, en spécifiant directement un attribut sans valeur associée dans une balise HTML. En XHTML, cette pratique est révolue. Dorénavant, afin de rendre votre code valide, il vous faut l'inscrire de manière complète, c'est-à-dire en spécifiant l'attribut et sa valeur associée, même si cela représente une répétition :

Invalide :

```
<option value="page.html" selected></option>
```

Valide :

```
<option value="page.html" selected="selected"></option>
```

L'attribut "name" est remplacé par l'attribut "id"

L'attribut "name", utilisé en HTML pour nommer les ancres, les images ou tout autre objet dans un document Web est remplacé par l'attribut "id" en XHTML. En effet, puisque le principe de nommer un objet revient à l'identifier et que par définition, cet identifiant se doit d'être unique, le recours à l'attribut "id" permet de s'assurer que la communication par le DOM avec un objet dans un document donné se fera individuellement. Malheureusement, le support pour l'attribut "id" étant faible ou inexistant dans les anciens navigateurs, il importe (en XHTML 1.0) de continuer à utiliser à la fois les attributs "name" et "id" pour désigner un même objet dans ces navigateurs, en leur attribuant des valeurs identiques, de sorte que les navigateurs de nouvelle génération puissent y trouver leur compte conformément aux règles de XML, tout en assurant une rétro-compatibilité avec les anciens navigateurs :

Invalide :

<h1 name="**titre**">**…**</h1>

Valide :

<h1 name="**titre**" id="**titre**">**…**</h1>

Ajoutons que l'attribut "name" des éléments a, applet, form, frame, iframe, img, et map est déprécié et ne peut plus être employé en XHTML [11].

Gestion des caractères spéciaux avec CDATA

XHTML est beaucoup plus sensible que ne l'était HTML aux caractères spéciaux dans les déclarations CSS et JavaScript. Vous ne pouvez plus inclure les blocs de code dans des balises de commentaires comme en HTML : en effet, les navigateurs supportant XML peuvent réagir de manière inattendue à la présence de ces caractères et simplement les ignorer, affichant ainsi le contenu des éléments script et style. Afin d'éviter un tel désastre, il est recommandé d'entourer les scripts et les styles d'une section CDATA, qui indiquera aux navigateurs XML que les caractères spéciaux inclus doivent être interprétés normalement.

```
<script language="javascript" type="text/javascript">
<![CDATA[Votre code javascript...]]>
</script>
```

Toutefois, cela ne règle que partiellement le problème puisque les navigateurs HTML ignorent le contenu d'une balise XML CDATA et requièrent l'utilisation traditionnelle des commentaires HTML. La seule solution viable à ce jour consiste donc à placer toutes les définitions de CSS ou de JavaScript dans des fichiers externes.

Gestion des caractères spéciaux dans les URL

Les caractères spéciaux présents dans les valeurs d'attributs s'avèrent également catastrophiques en XHTML. Afin de contrer le problème, vous devez nécessairement les encoder afin d'éviter que le navigateur ne les interprète de façon erronée. Ainsi, pour tous les caractères spéciaux comme « < », « > » ou « & » destinés à être interprétés tels quels, vous devrez plutôt inscrire « < », « > » ou « & » :

Invalide :

```
<a href="index.php?a=1&b=2" title="Articles & Didacticiels">
```

Valide :

```
<a href="index.php?a=1&amp;b=2" title="Articles &amp;amp; Didacticiels">
```

Conformité et type de document

Afin d'être conforme aux normes XHTML, outre les spécificités de syntaxe XML décrites précédemment, un document doit respecter les règles suivantes :

Prologue XML et encodage de caractères

La déclaration XML <?xml?> est une composante recommandée du document XHTML. Cette déclaration l'identifie en effet comme appartenant au cadre XML et en décrit la version.

Son support inégal dans les principaux navigateurs et ses conséquences sur le rendu CSS incitent parfois à l'omettre.

Cependant, comme elle permet de spécifier l'encodage de caractères spéciaux dans le document, choisir de l'omettre expose à un rendu incorrect de ces mêmes caractères. Le cas typique est celui des documents rédigés en français, qui utilisent des caractères spéciaux n'appartenant pas à l'encodage ASCII [10].

Tout dépend donc de l'encodage choisi :

- avec les encodages par défaut du XML (*UTF-8* et *UTF-16*), le prologue XML est optionnel, et une balise `meta` précisera l'encodage approprié pour les anciens navigateurs : `<meta http-equiv="content-type" content="text/html; charset=UTF-8" />`;
- avec d'autres encodages (ISO-8859-1 par exemple), il faut :
    - soit inclure en tête du document le prologue : `<?xml version="1.0" encoding="iso-8859-1"?>` ainsi que la balise `meta` pour les vieux navigateurs qui ignoreront ce prologue ;
    - ou encore spécifier l'encodage au niveau supérieur, c'est-à-dire dans l'en-tête HTTP serveur.

Utilisation de la déclaration de type de document

Le document doit respecter les normes de validation d'une des trois DTD XHTML : strict, transitional ou frameset. Une déclaration de type de document doit apparaître dans le document juste avant l'élément html (qui est l'élément racine de tout document XHTML) [11].

- La DTD stricte n'autorise pas l'utilisation des anciens éléments de présentation (s, u, etc.) mais elle a l'avantage de contraindre le développeur à séparer structure et présentation, avec les facilités de maintenance que cela apporte.
- La DTD transitionnelle est plus permissive et plus proche des anciennes habitudes liées à HTML 3.2. Les balises dépréciées en XHTML strict y sont autorisées : le recours à cette DTD est donc plus facile dans un premier temps, mais avec le défaut de mêler encore partiellement structure et présentation.
- Enfin, la DTD frameset permet l'utilisation des cadres. Ceux-ci tombent en désuétude, mais peuvent se réveler encore utiles dans certains cas très exceptionnels.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

5. Langage XML

XML est une recommandation du World Wide Consortium (W3C) depuis février 1998 (révision mineure le 6 octobre 2000). Cette norme est simple, et c'est cette simplicité qui fait sa puissance.

XML est en fait un **langage** permettant de définir des **formats de documents** et de créer des documents respectant ces formats. Ainsi, et c'est important, **XML n'est pas un format de document** comme tel.

XML est un *méta-langage* à *balises*.

- Un *méta-langage*: il sert à définir des langages
- A balises: il est composé de blocs de textes structurés par des balises de début et de fin, par ex. <nom>toto</nom>

5.1 Documents XML

Un document est appelé "document XML" s'il est *bien formé*, ce qui signifie qu'il respecte les règles syntaxiques de XML. Il est *valide* si, en plus, il respecte la grammaire du langage, contenue dans un fichier appelé DTD (Définition de Type de Document).

Physiquement, un document XML est composé d'unités qu'on appelle *entités*, qui peuvent appeler d'autres entités de façon à ce qu'elles soient aussi inclues dans le document.

Du point de vue de la structure logique, le document est composé explicitement de :

- déclaration de document XML
  <?xml version="1.0" encoding="ISO-8859-1"?>
- déclaration de type de document (DTD)
  <!DOCTYPE AML SYSTEM "AML.dtd">
- éléments (certains avec des attributs)
  <IDENT>UGC 6</IDENT> ou
 (ce dernier est un élément vide)
- commentaires
  <!-- par exemple, <élément-non-traité/> -->
- entités référence ou entités de caractères
  &copyright; ou ’
- instructions de traitement (processing instructions)
  <?name pidata?>
- sections littérales
  <![CDATA[*p=&q; b=(i<=3);]]>

5.2 Schémas XML

- Les *schémas* XML ont surtout été créés pour pouvoir définir un langage XML avec la syntaxe XML
- Les schémas XML définis par le W3C (WXS, W3C XML Schema) permettent de spécifier beaucoup plus de contraintes que les DTD, en particulier pour les types de données
- Ils sont très complexes, ce qui a entraîné la création de RELAX NG, un autre langage de schéma, par des organisations indépendantes du W3C [10]

Exemple d'un schéma XML :

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="personne">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="nom" type="xs:string" />
          <xs:element name="prenom" type="xs:string" />
          <xs:element name="date_naissance" type="xs:date" />
          <xs:element name="etablissement" type="xs:string" />
          <xs:element name="num_tel" type="xs:string" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

5.3 Le standard XLink

XLink est un standard utilisant la syntaxe XML pour définir des liens entre des ressources. Il permet non seulement de définir des liens simples comme les liens HTML, mais aussi, entre autres :

- de donner du sens aux liens
- de leur donner un titre
- de les utiliser dans les deux sens
- de lier plus de deux ressources avec un seul lien
- de préciser le comportement à suivre quand on suit un lien

L'intérêt de XLink est de permettre une compréhension des liens entre des documents XML, quelque soit le langage XML utilisé. En effet, chaque langage XML peut utiliser des

noms différents pour représenter des liens, mais cela rend difficile l'utilisation de liens dans un ensemble de documents XML utilisant des langages différents. Avec une syntaxe commune, il devient possible de faire des traitements automatiques utilisant les liens et leur sens, ou par exemple de gérer ce qui se passe quand on clique sur un lien dans un navigateur XML. [10]

```
<personne
xlink:href="etudiants/patjones62.xml"
xlink:label="etudiant62"
xlink:role="http://www.exemple.com/props_lien/etudiant"
xlink:title="Pat Jones" />

<personne
xlink:href="profs/jaysmith7.xml"
xlink:label="prof7"
xlink:role="http://www.exemple.com/props_lien/professeur"
xlink:title="Dr. Jay Smith" />

<cours
xlink:href="cours/cs101.xml"
xlink:label="SI-101"
xlink:title="Sciences Informatiques 101" />
```

6. Conclusion

Après avoir fait un survol sur les notions fondamentales des sites web et de ses langages, à savoir le HTML, le XHTML et le XML, nous pouvons en ce moment passer à la description et à la conception de notre projet. Ce que nous allons le voir dans le chapitre suivant.

Chapitre III

Conception de l'application

Chapitre III

Conception de l'application

1. Introduction

La conception demeure la tâche la plus fastidieuse du cycle de vie d'un projet. Ceci est principalement dû aux efforts de raisonnement, d'abstraction et d'imagination qu'il faut effectuer pour modéliser et concevoir le futur système.

Notre application se veut être un outil que l'informaticien doit utiliser pour l'aider à la conception de la base de données issue d'un site web dynamique.

Nous décrirons le long de ce chapitre les étapes nécessaires à la réalisation de cet objectif.

2. Choix des pages déléguées dans un site :

Après avoir choisi un site web dont on veut reconstruire la base de données, nous devrions regrouper les pages de ce site selon la même forme de données.

Ce regroupement est effectué d'une façon manuelle. Le concepteur, à savoir l'informaticien, devra réaliser ce regroupement en consultant toutes les pages du site une à une. Les pages ayant un ensemble de données semblables doivent constituer un groupe.

Durant toutes les étapes de notre projet, nous prenons comme exemple, un site contenant les informations des étudiants et des enseignants (voir Figure 3.1 relative à une page d'un enseignant).

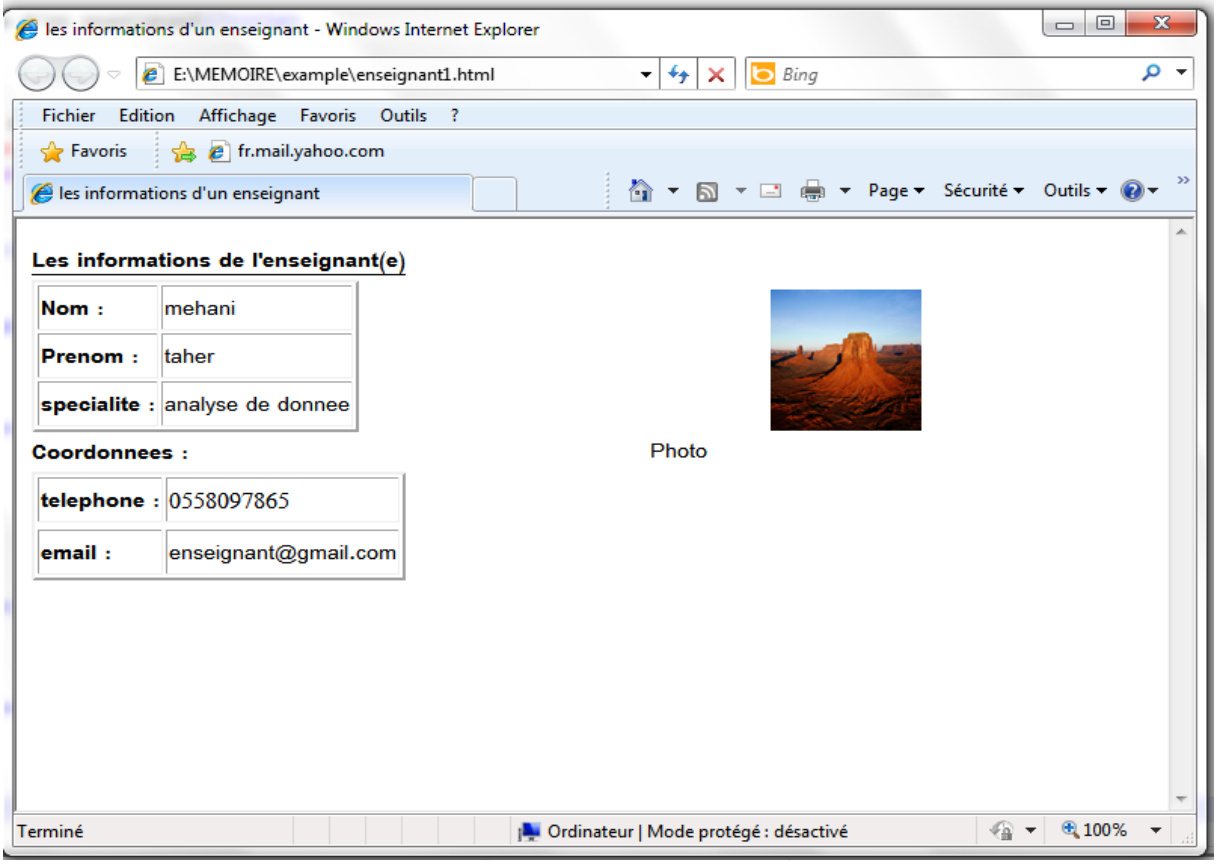


Figure. 3.1 Page d'un enseignant

Une consultation rapide de ce site permet de regrouper les pages contenant les informations des étudiants en un même groupe et celles qui contiennent les informations des enseignants en un autre groupe.

Pour chaque groupe constitué, on sélectionne une page, qu'on appellera page déléguée, qui doit contenir le maximum d'informations. L'étape suivante de notre tâche sera réalisée sur la page déléguée.

3. Transformation de la page HTML en XHTML :

Les fichiers html ne sont pas fortement structurés, c'est pour cette raison que les navigateurs s'efforcent à minimiser les erreurs présentes dans une page html pour l'afficher d'une manière acceptable.

Pour ne pas prendre le risque de gérer les pages html avec leurs erreurs de syntaxe et de structure, une étape de prétraitement est nécessaire sur le code source de la page.

Lors de ce prétraitement, nous ne nous bornerons pas à corriger les erreurs, mais nous en profitons pour faire des sources HTML des fichiers bien formés au sens XHTML. Ce travail est effectué grâce à une transformation du fichier html à un fichier XHTML en utilisant les outils de transformation existants.

La figure 3.2 donne une représentation arborescente de la page html enseignant (la page déléguée sélectionnée lors de la phase précédente) transformée en XHTML.

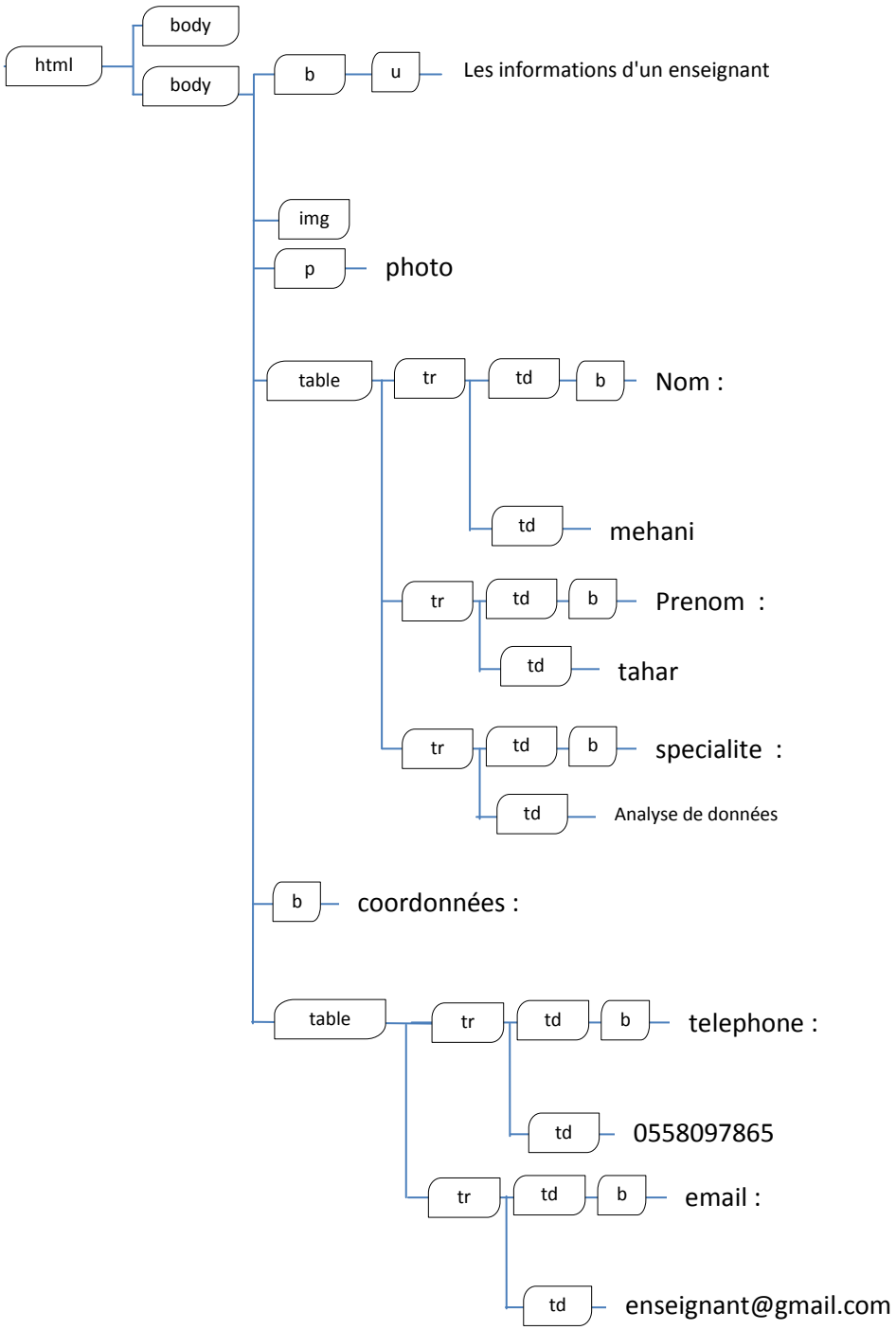


Figure. 3.2 Page déléguée enseignant transformée en XHTML.

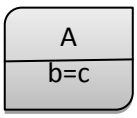
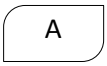
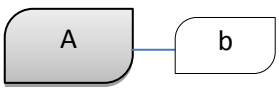
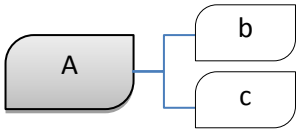
4. Phase de création du squelette

Le langage HTML est très pauvre en sémantique. Les informations qu'on peut en tirer sont insuffisantes pour retrouver la structure sémantique d'un site. Il nous faut donc ajouter au code HTML des informations structurelles et sémantiques.

Pour se faire, nous allons, pour chaque page déléguée, définir une grammaire que nous appelons Squelette qui prendra la forme d'un fichier XML.

Le fichier Squelette nomme et hiérarchise l'ensemble des concepts identifiés dans la page déléguée. A cette fin, on utilise un formalisme composé d'éléments XML où l'on intègre également le balisage HTML du fichier analysé afin de localiser précisément les concepts. Le fichier Squelette créé va servir de modèle pour extraire les données de toutes les pages html du groupe de la page déléguée.

Ci-dessous, nous décrivons le formalisme du squelette en introduisant l'un après l'autre chaque concept et attribut XML utilisé. Pour représenter les structures des fichiers html ou XML, nous utilisons les symboles graphiques illustrés dans le tableau suivant (Tableau 3.1).

Symbole graphique	Description
	Elément réservé de nom « A » contenant un attribut de nom «b» dont la valeur est «c»
	Elément réservé de nom « A » ne contenant pas d'attribut
	Elément HTML de nom « A »
Abc	Constante textuelle
	L'élément de nom « A » contient uniquement un élément de nom « b»
	L'élément de nom «A » contient un élément de nom « b » suivi d'un élément de nom «c»

Tab. 3.1 Description des symboles graphiques utilisés

Pour illustrer notre propos, nous reprenons l'exemple du site des enseignants et étudiants, en spécifiant le traitement de la page déléguée du groupe des enseignants.

4.1 Les informations structurelles du squelette :

Comme nous l'avons signalé précédemment, le fichier squelette contient les éléments Html en leur ajoutant des éléments spéciaux qui n'existaient pas et n'apparaissaient pas dans le balisage de l'Html, mais qui devront enrichir le fichier par des informations concernant la structure des données présentes dans la page html.

Ces nouvelles informations structurelles sont, en fait, relatives à la structure de la page du point de vue conceptuel, et vont servir à la détermination future de la structure même de la table de la base de données.

Dans ce qui suit, nous détaillons l'ensemble de ces notions, en prenant toujours comme exemple celui mentionné précédemment, et en particulier la page déléguée des enseignants.

➢ Notion de *Root* : racine du fichier Squelette

Tout fichier XML bien formé, doit avoir une racine unique. Par convention, elle est identique dans tous les fichiers squelettes et prend la forme de l'élément Root (Figure 3.3)[4].

Figure. 3.3 Représentation graphique de l'élément *Root*

➢ Notion de *metaelement* :

Cet élément sert à définir le domaine (ou la notion) auquel appartient la donnée.
Le nom de ce domaine est, dans le contexte des bases de données, appelé champs.

Nous utilisons l'élément metaelement pour donner un nom de champs relatif à la donnée en cours, ou faire référence à une autre. Pour faire différencier les deux usages, cet élément est toujours accompagné d'un et un seul des deux attributs suivants :

- **Name** : Chaque notion identifiée dans le fichier squelette est représentée par un <metaelement> pourvu d'un attribut Name qui permet de lui donner un nom représentatif. On le déclare comme fils direct de la racine du document. Ainsi, l'ensemble des fils directs de la racine constitue la liste de toutes les notions présentes dans la page.

Dans notre exemple, un coup d'œil à la page déléguée (fichier XHTML) nous permet de découvrir les notions suivantes: Enseignant, Nom, Prénom, Photo, spécialité, coordonnées, téléphone et email (voir Figure 3.4). Ces metaelement sont ajoutés au fur et à mesure au fichier XML. [4]

Notons un cas particulier dans la déclaration des notions : on déclare comme premier fils de la racine une notion principale qui représente la globalité des informations contenues dans la page. Dans notre cas, le premier élément de la liste est donc la notion Enseignant.

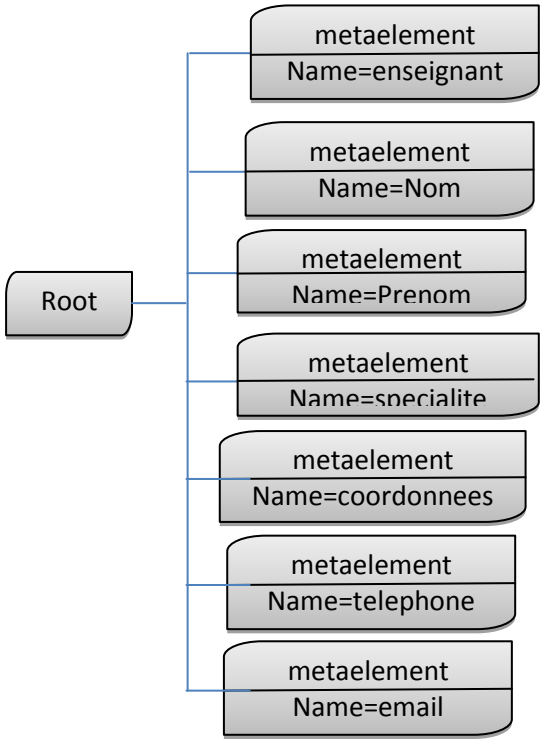


Figure. 3.4 Arbre des notions de la page déléguée

- **Ref** : Hormis la notion principale qui est représentée par metaelement, chaque élément déclaré peut faire partie de la définition d'un autre élément (dans notre exemple, les notions Telephone et email font partie de la notion Coordonnees).

Pour représenter ce phénomène, on insère dans la définition de la notion englobant (Coordonnées dans notre exemple) un <metaelement> pour chaque sous-notion (Telephone et email). On donne à chacun un attribut Ref dont la valeur est le nom de la notion qu'il référence.

Par conséquent, le domaine de valeur de l'attribut Ref dans un fichier squelette est inclus dans le domaine de valeur de l'attribut Name dans le même fichier. L'utilisation du mécanisme de référence a pour effet d'importer la totalité de la déclaration d'une notion au sein d'une autre.

La figure 3.5 donne la représentation arborescente de la notion coordonnées englobant les sous-notions téléphone et email.

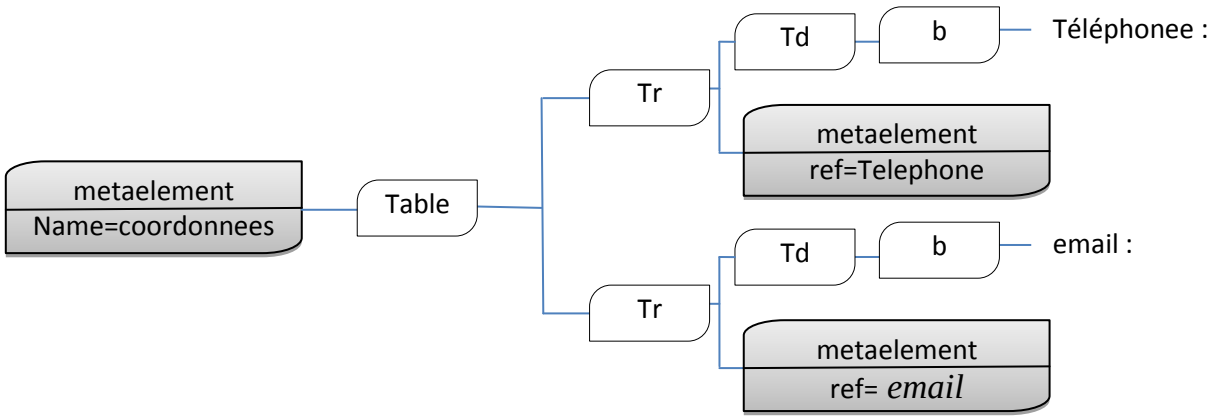


Figure. 3.5 - Notion de Coordonnées est composé des deux sous-notions Téléphone et email.

➢ Notion de metavalue :

Cet élément <metavalue> sert à définir les données dans le fichier Html, ce qu'on veut extraire les données, quant à elles, doivent être localisées précisément afin de pouvoir être extraites lors d'une étape ultérieure.

Afin de représenter ces données, nous introduisons un nouvel élément XML propre à notre formalisme : l'élément <metavalue>. Etant donné que notre objectif est d'attribuer une valeur sémantique aux informations à extraire.[4]

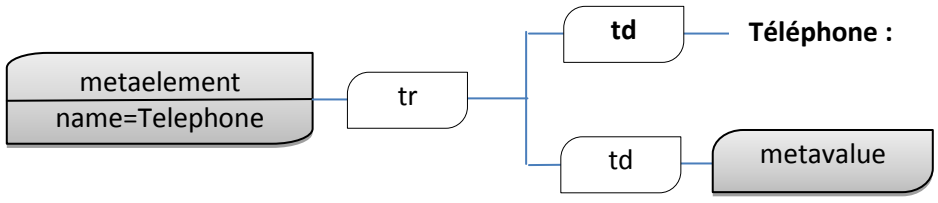


Figure. 3.6 L'élément Téléphone est placé dans une ligne de tableau. La première cellule contient une Constante textuelle («Téléphone :») formatée en gras et la seconde une donnée variable.

➢ Notion de *metavalueattributs* :

Comme nous l'avons noté plus haut, la plupart des attributs utilisés dans le balisage HTML ne servent qu'à préciser le formatage (couleur de fond, couleur du texte, taille d'une cellule de tableau, etc.). Cependant, la valeur de certains attributs représente une donnée pertinente à importer dans notre future base de données. C'est le cas notamment de l'attribut src de la balise <img/> qui renseigne l'adresse où est enregistrée l'image à afficher.

Pour localiser ce type de données nous affectons à l'élément <metavalueattributs> deux attributs, un attribut ref pour référencier le <metaelement> qui contient l'élément HTML (<img/>), et un attribut nameAttribut on lui donne comme valeur le nom de l'attribut qui nous intéresse dans le code source (src pour une image par exemple); l'élément HTML qui contient l'attribut en question devient le fils de l'élément <metaelement>. Dans notre exemple, la notion Photo illustre ce cas. (Figure 3.7).

Figure. 3.7 Une image est représentée par deux éléments : photo qui référencie le metaelement et src qui donne l'emplacement de l'image. [4]

➢ Notion de *metagroup* :

L'élément XML < metagroup> sert à faire de répéter le contenu de concept group, cet élément est toujours accompagné de deux attributs min et max. Les attributs min et max sont utilisés pour représenter deux cas particuliers dans le mécanisme de référence, à savoir son aspect facultatif et/ou répétitif.

D'une part, il arrive qu'un sous-élément d'un concept ne soit présent que dans certaines pages du type (sous-élément optionnel ou facultatif). Dans notre exemple, nous pouvons imaginer que certaines pages ne mentionnent pas de numéro de téléphone. On note cette particularité en donnant la valeur 0 à l'attribut min de concept de metagroup qui contient la référence au concept Telephone.

D'autre part, il arrive qu'un même sous-élément ait plusieurs instances successives au sein d'un concept (sous-élément multivalué ou répétitif). Dans notre exemple, nous pouvons imaginer que certains enseignants ont deux numéros de téléphone.

On utilise l'attribut max pour représenter cette particularité. Sa valeur précise le nombre maximum d'instances successives d'un même sous-concept. Dans le cas où ce

nombre est indéterminé, l'attribut prend la valeur spéciale N. Ajoutons que la valeur de min doit toujours être inférieure ou égale à celle de max.

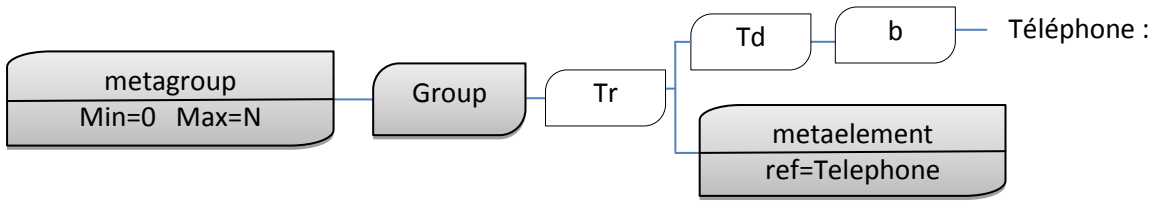


Figure. 3.8 Le sous-élément Téléphone de la notion Coordonnées est facultatif et multivalué

➢ Notion de *metaany* :

Souvent des parties de pages sont dépourvues d'informations intéressantes relatives au domaine d'application. C'est typiquement le cas d'une bannière publicitaire de navigation intrapage.

Afin d'éviter de recopier inutilement dans le fichier squelette des portions de code source dénuées d'intérêt, nous introduisons un nouvel élément XML dans notre formalisme :<metaany/>. Cet élément se place comme fils unique d'un élément HTML et fait office de joker (ou wild-card). Il signale que cet élément peut contenir n'importe quoi et que lui et ses descendants peuvent être ignorés. Etant donné que cet élément remplace le contenu d'un élément, il n'a jamais de frères ou de fils. [4]

Dans notre exemple, nous utilisons ce mécanisme pour cacher la bannière en haut de la page (Figure 3.9).

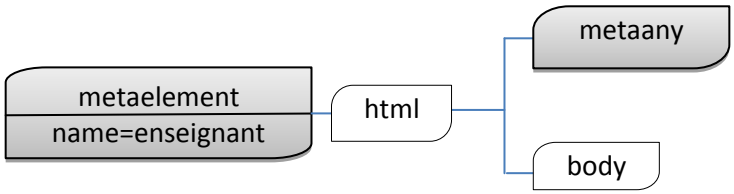


Fig. 3.9 - <*metaany*> remplace le contenu de l'élément HTML *head* qui renferme la tête de fichier Html.

➢ Notion de *metachoice*

Rappelons qu'un fichier squelette doit décrire toutes les pages d'un même groupe et non une seule instance. Par conséquent, si la structure ou la présentation d'une notion

change d'une page à l'autre, il doit tenir en compte les différences qui peuvent être de plusieurs natures. Nous en donnons la liste ci-dessous et décrivons le traitement qu'il convient de leur appliquer.

- **Différence de présentation d'un élément :**

Certains éléments d'une page d'un groupe peuvent être affichés de manière différente d'une page à l'autre. Imaginons dans notre exemple que la constante de texte « nom : » soit formatée tantôt en gras, tantôt en italique. Afin de représenter ces différences, nous ajoutons deux nouveaux éléments XML à notre formalisme.

L'élément <metachoice> se positionne comme premier fils dans la déclaration d'un élément. Il signale que la notion a différentes descriptions possibles. Chacune de ces descriptions prend place dans un élément <choice > qui obéit aux règles de construction propres à la déclaration des notions. L'ensemble des éléments <choice> deviennent les fils de l'élément <metachoice>.

La figure 3.10 illustre la structure de choix pour la constante de texte « nom : ».

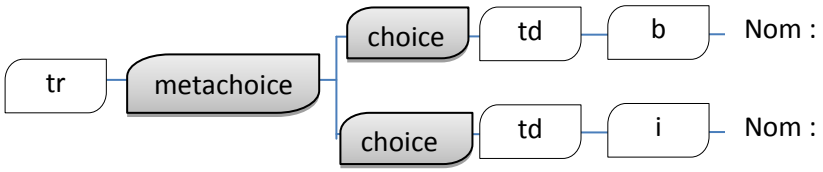


Figure. 3.10 Représentation d'une structure de choix pour la constante textuelle « nom : »

- **Différence d'ordre des sous-éléments d'une notion :**

Une différence se manifeste également lorsque l'ordre d'apparition des sous-notions d'un élément varie d'une page à l'autre. Dans notre exemple, on peut imaginer que la notion email apparaisse avant la notion Téléphone dans certaines pages. C'est à nouveau la structure de choix qui nous permet de représenter ce phénomène.

- **Différence de contenu sémantique d'une notion :**

Parfois, une même notion présente un contenu sémantique différent d'une page à l'autre. Ce serait le cas dans notre exemple si la notion Coordonnées contenait soit un numéro de téléphone, soit un Email, mais pas les deux. Pour représenter ce type de cas, nous utilisons à nouveau la structure de choix.

4.2 Représentation structurelle d'un fichier squelette :

Nous donnons par la figure 3.11, une représentation générale du fichier squelette relatif à la page déléguée enseignant.

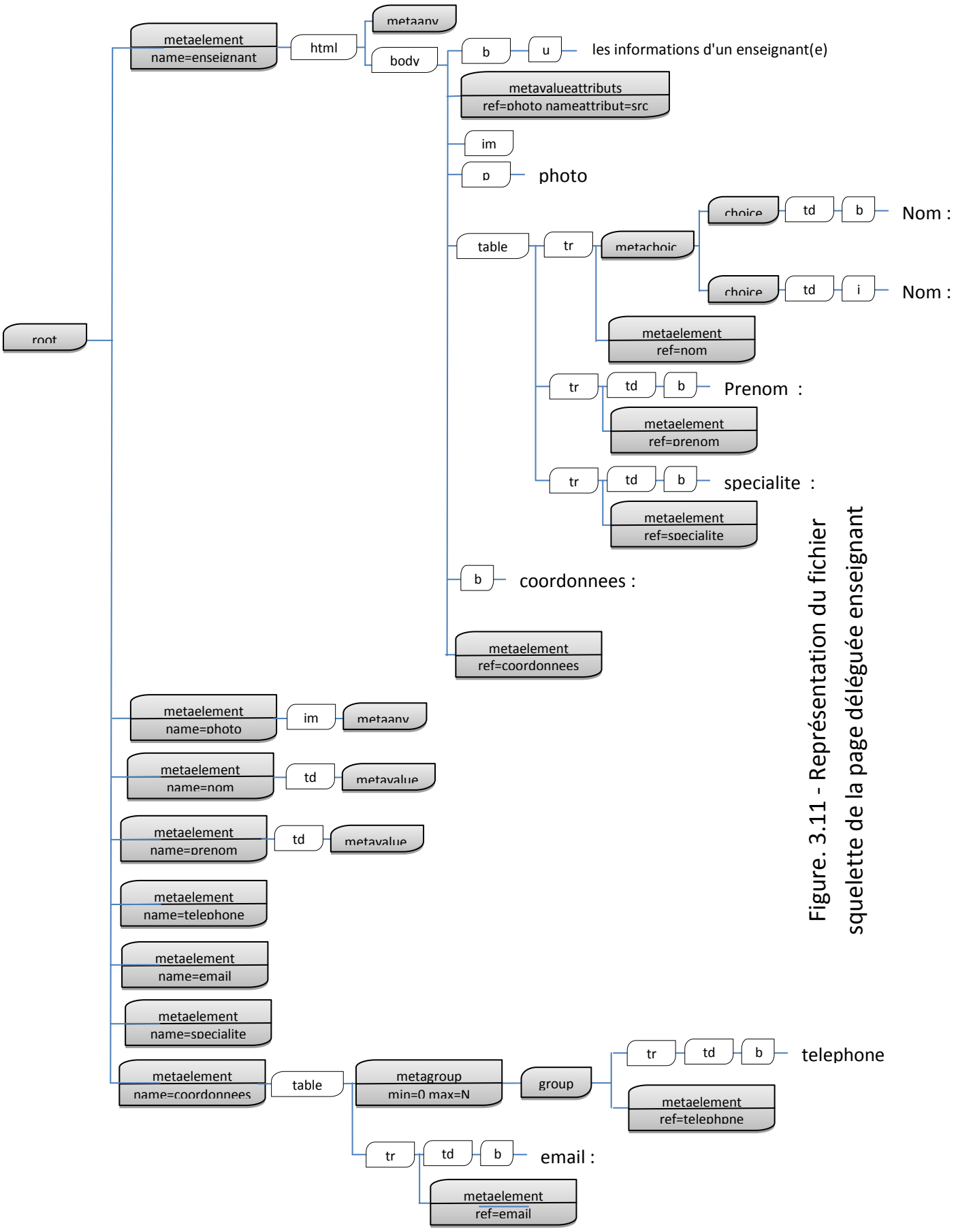


Figure. 3.11 - Représentation du fichier squelette de la page déléguée enseignant

5. Phase d'extraction des données:

Après la phase de création du squelette d'une page html déléguée d'un groupe de pages du site web, nous passons à la phase d'extraction des données depuis les pages html du groupe.

Cette phase consiste à prendre, une par une, toutes les pages html du groupe afin de les traiter. Pour chaque page, toutes ses données relatives aux structures présentes dans le squelette sont extraites ensuite mises dans un fichier XML.

Le résultat final de l'extraction de toutes les pages est un fichier XML contenant la structure et les données relatives aux pages html du groupe. Pour notre exemple, la figure 3.12 donne le contenu du fichier XML présentant le résultat d'extraction de la page déléguée enseignant.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <enseignant>
    <photo>Desert.jpg</photo>
    <nom>mehani</nom>
    <prenom>taher</prenom>
    <specialite>analyse de donnee</specialite>
    <coordonnees>
      <group>
        <telephone>0558097865</telephone>
      </group>
      <email>enseignant@gmail.com</email>
    </coordonnees>
  </enseignant>
  <enseignant>
    <photo>N95591.jpg</photo>
    <nom>moussaoui</nom>
    <prenom>Adel</prenom>
    <specialite>systeme d'Exploitation</specialite>
    <coordonnees>
      <email>enseignant2@gmail.com</email>
    </coordonnees>
  </enseignant>
</root>
```

Figure. 3.12 Fichier XML résultat d'extraction de la page déléguée enseignant

Ce fichier peut être considéré comme une table de la base de données extraite à partir des pages html. L'informaticien ayant en sa possession le fichier XML, peut compléter la phase de rétro-conception de la table, en définissant les types des champs, leur dimension, ainsi que la clé primaire.

6. Phase de rétro-conception de la base de données

En répétant les étapes 2 à 5, pour chaque groupe de pages constitué, on peut aboutir à un ensemble de fichiers XML, contenant la structure et les données relatives à chaque groupe de pages, et constituant l'ensemble des tables formant la base de données.

7. Conclusion

Ce chapitre a illustré les différentes phases nécessaires pour atteindre l'objectif du projet. En débutant par une classification (regroupement) des pages html et en sélectionnant pour chaque groupe, une page déléguée qui va être transformée en XHTML, vient la phase cruciale qui consiste à construire un squelette XML contenant les définitions des structures utilisées. L'étape suivante est l'extraction des données dans un fichier XML, à partir de toutes les pages html du groupe. Enfin, une synthèse des fichiers XML construits à partir de tous les groupes de pages du site, permet d'élaborer un schéma de la base de données.

Nous décrirons dans le chapitre suivant les éléments essentiels relatifs à l'implémentation de l'application.

Chapitre IV

Implémentation de l'application

Chapitre IV
Implémentation de l'application

1. Introduction

Notre travail consiste à réaliser un système d'extraction des données à partir des pages html d'un site dynamique, dans l'objectif de reconstituer la base de données avec laquelle ce site fonctionne.

Pour réaliser cet objectif, nous avons décomposé le problème en trois phases principales :

- Regrouper les pages du site web et choisir de chaque groupe une page déléguée.
- Créer un fichier squelette de chaque groupe à partir de sa page déléguée.
- Prendre le fichier squelette ainsi que les pages de groupe concerné puis effectuer l'opération d'extraction des données des pages pour construire un fichier XML englobant l'ensemble des structures et des données relatives au groupe.

Pour cela, nous avons réalisé de grands efforts pour implémenter le système. Le reste du chapitre sera consacré à la description de l'architecture ainsi que les fonctionnalités et les différentes procédures du système. Ensuite, nous décrivons en détail les différentes tâches du système sur un exemple de site dynamique.

2. Environnement de développement

2.1 Langage de programmation :

Nous avons utilisé C# comme langage de programmation orienté objet. Nous avons choisi l'orienté objet pour la facilité de développement, la maintenance et la réutilisation des différents modules de l'application[8],[9].

2.2 Les bibliothèques utilisées :

Nous avons utilisé deux bibliothèques qui sont :

- System.dll: Cette bibliothèque contient le package « XML » qui contient la classe « XmlDocument », elle est utilisée pour exploiter les fichiers XML[8].
- Html2Xhtml.dll: Cette bibliothèque contient le package « Corsis.Xhtml» qui contient la classe « Html2Xhtml », elle est utilisée pour convertir le fichier html en un fichier xhtml.[12]

3. Architecture du système

L'architecture générale de notre système est composée des modules suivants :

- **Le module de préparation** : qui consiste à regrouper les pages html homogènes, ensuite sélectionner pour chaque groupe une page élite (déléguée) et la convertir en xhtml.
- **Le créateur de squelette** : qui consiste à construire un fichier XML englobant les différentes structures des données présentes dans la page déléguée.
- **L'extracteur** : qui consiste à extraire les données depuis les pages html du groupe et les mettre dans un fichier XML.

La figure 4.1 illustre les fonctionnalités fondamentales du système que nous avons réalisé.

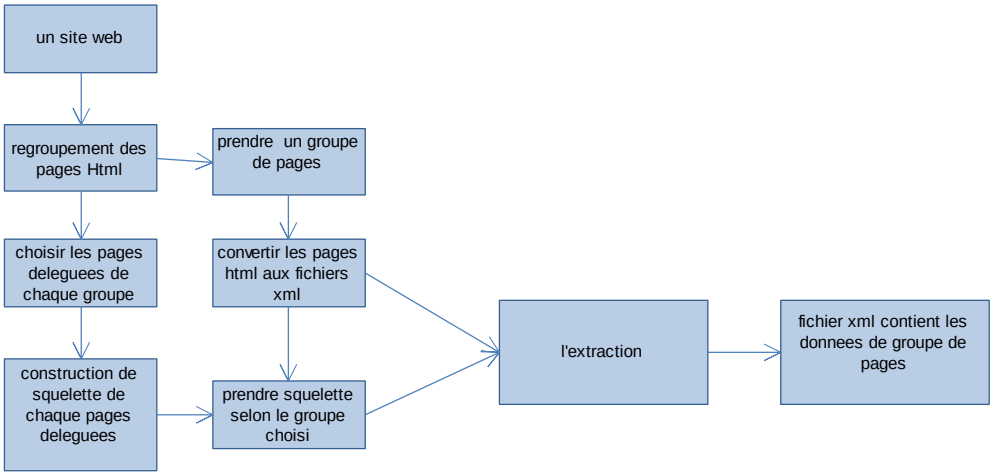


Figure 4.1 Fonctionnement du système d'extraction

4. Le module : Créateur de squelette

Pour construire le squelette d'un groupe de pages, il faut charger la page déléguée, ensuite effectuer un traitement guidé par l'utilisateur, pour aboutir à la définition des concepts (notions) fondamentaux existants dans la page. Le résultat est sauvegardé dans un fichier XML.

Le déroulement de cette procédure est illustré dans l'organigramme suivant (Figure 4.2).

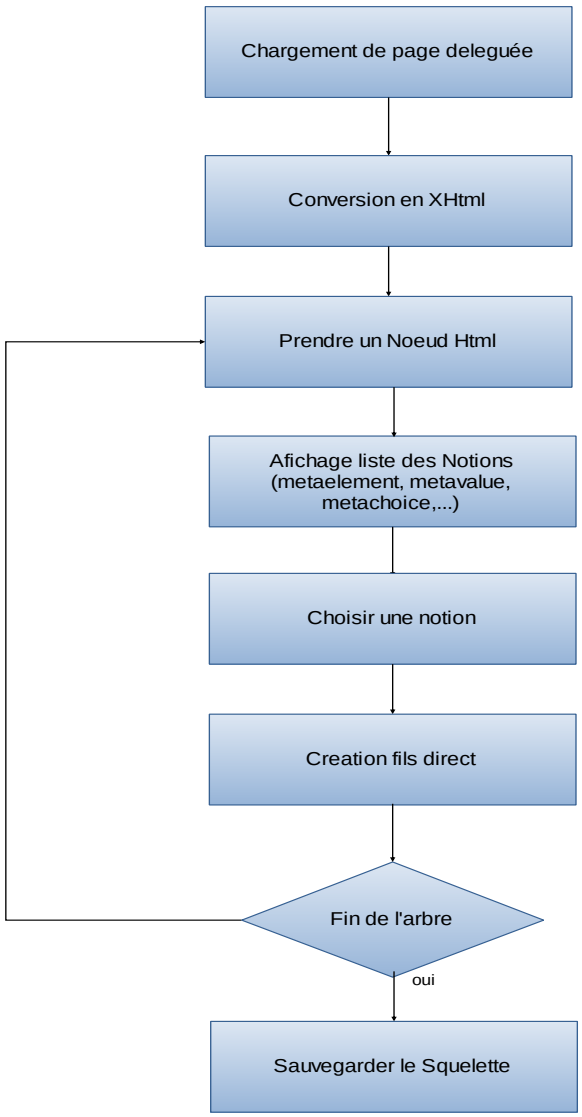


Figure 4.2 Organigramme du créateur de squelette

5. Le module : Extracteur de données

Le module d'extraction des données sert à extraire les données des pages html du groupe, et à l'aide du squelette créé, il effectue une comparaison de chaque nœud html et chaque nœud du squelette, pour localiser les différentes structures issues du squelette et les associer avec leurs données correspondantes issues de la page html.

Le déroulement de cette procédure est illustré dans l'organigramme suivant (Figure 4.3).

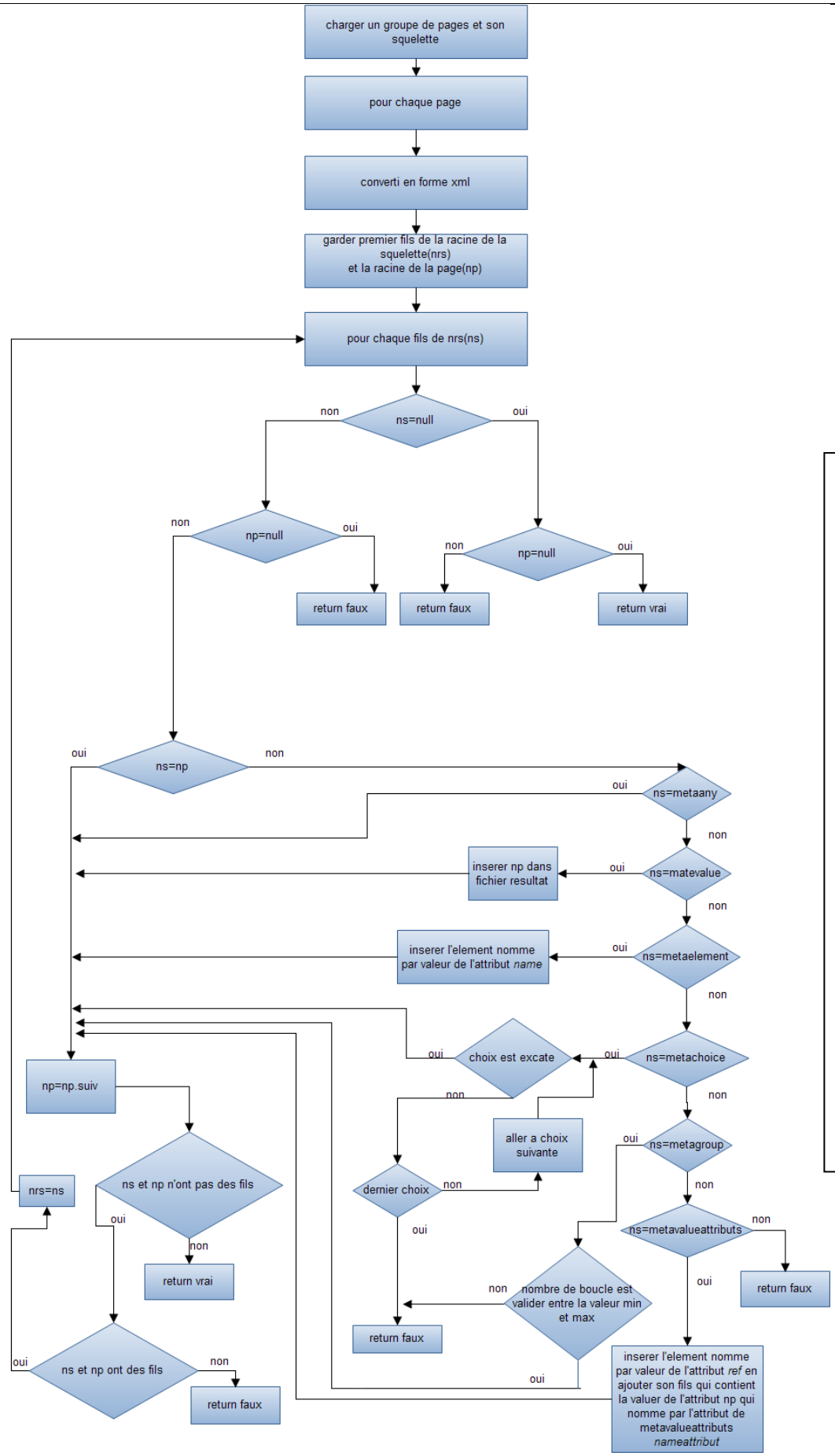


Figure 4.3 Organigramme de l'extracteur de données

6. Exécution de l'application sur un exemple

Dans ce qui suit, nous allons présenter les différentes interfaces relatives au fonctionnement de l'application sur un exemple de site dynamique détaillé.

Nous reprenons l'exemple du site des enseignants et étudiants. Ce site est constitué de plusieurs pages d'enseignants. Chaque enseignant assure le tutorat de plusieurs étudiants. Nous allons détailler le déroulement de l'exécution sur une page déléguée des enseignants, ensuite nous traitons toutes les pages du groupe enseignant.

6.1. Fenêtre principale

C'est la fenêtre principale de l'application qui permet d'accéder aux autres fenêtres. Elle est composée d'un menu principal, d'une barre d'outil et d'une zone d'affichage des fenêtres MDI filles (Figure 4.4)

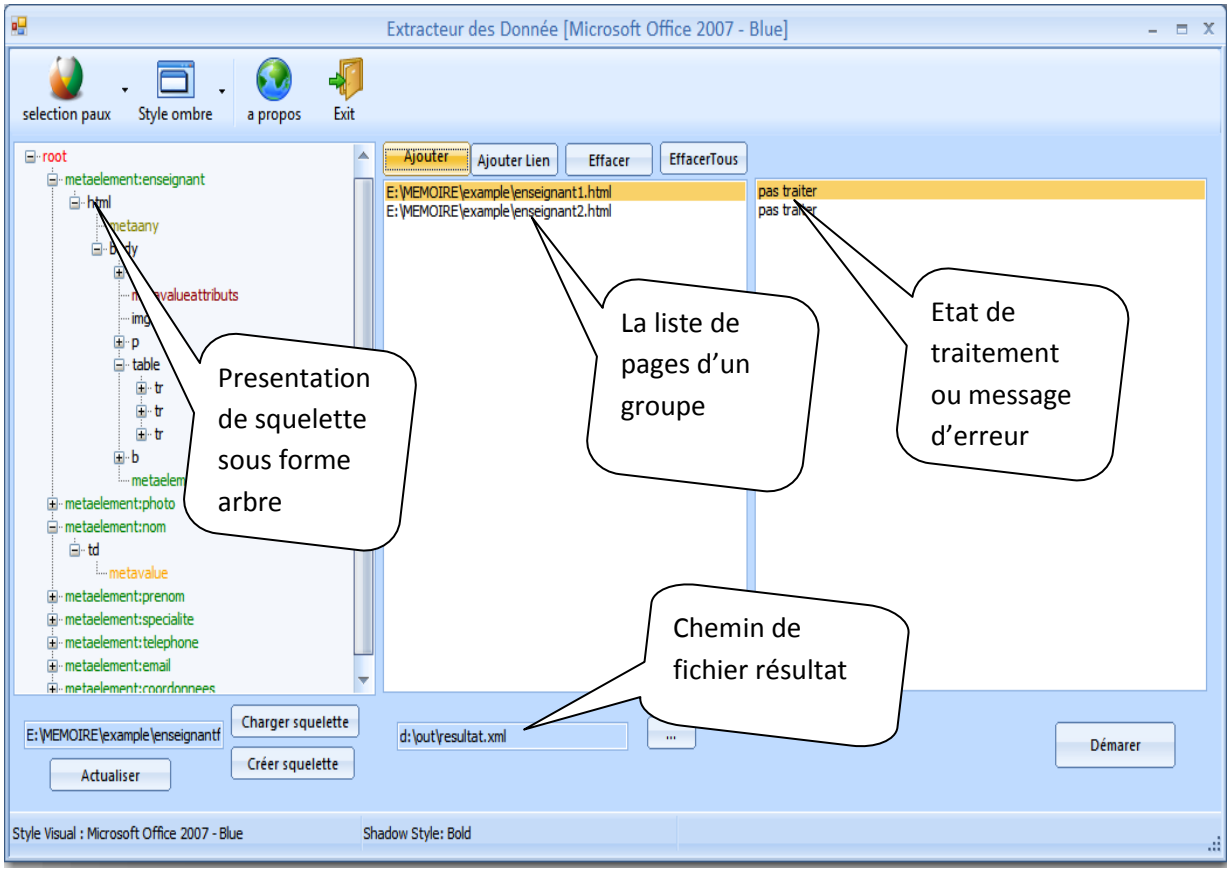


Figure 4.4 Fenêtre principale de l'application

Pour notre exemple, nous allons créer un nouveau projet de squelette, en cliquant sur le bouton « Créer squelette ». La fenêtre suivante apparaît.

6.2. Fenêtre de Création du Squelette :

Cette fenêtre contient six boutons d'exploitation et un menu plus détaillant en commence par le plus à gauche :

- Permet de prendre un lien d'une page internet et la rendre sous forme arbre.
- Créer un nouveau projet à partir d'une page déléguée et la rendre sous forme d'un arbre.
- Ouvrir un projet existant pour faire des modifications sur le squelette.
- Sauvegarder le projet dans un chemin.
- Sauvegarder le squelette dans un chemin.
- Sauvegarder le texte modifié dans une notion.
- Pointer sur un nœud de squelette et cliquer à droite pour ajouter des notions comme se présente dans la figure 4.5

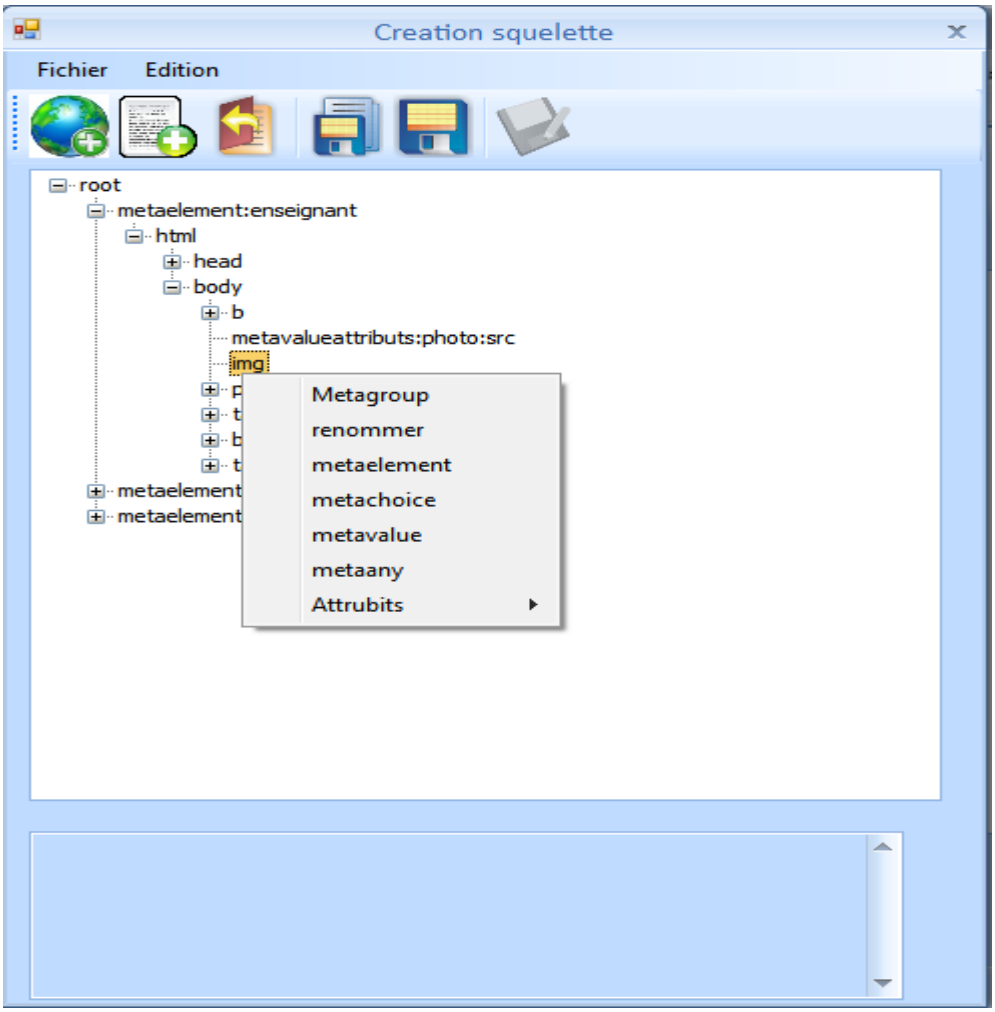


Figure. 4.5 Fenêtre de création de squelette de l'application

Le déroulement de création du squelette est détaillé dans ce qui suit :

1- Création de nœud Enseignant qui sera le premier fils du Root qui contient l'arbre Html de la page.
2- En développant le Html fils d'enseignant on trouvant le Head et le nœud Body
3- On pointe vers le head en cliquant à droite et créer metaany voir fig.4.5 qui permet d'ignorer tel nœud.
4- On développe le nœud body et en pointe vers Img en choisissant attributs qui contient lui-même plusieurs. On clique sur source et en lui donne le nom de champ « photo » qui sera ajouté comme fils direct<metaelement:photo >du Root.
5- On développe le nœud table et le premier fils tr qui contient deux nœuds td eux même contiennent des fils. Nous le parcourons jusqu'au #texte qui contient une information intéressante comme « Nom :» si c'est une notion peut être présente sous différentes formes dans les pages, une en italique et d'autre en bold (*Nom*, **Nom**) ou bien (Nom :,Nom /) …etc. On pointe vers le nœud père puis on clique sur metachoice en lui donnant le nombre de cas possibles. Le système va créer la même branche avec le même nombre (voir figure 4.6). Si ce n'est pas une notion mais une valeur d'une notion, on pointe sur elle en cliquant metavalue en montant vers leurs père en créent un metaelement en lui donnant un nom de champs qui va porter la valeur de ce metaelement et sera ajouté comme fils directe <metaelement:Nom> du Root (voir figure 4.7).

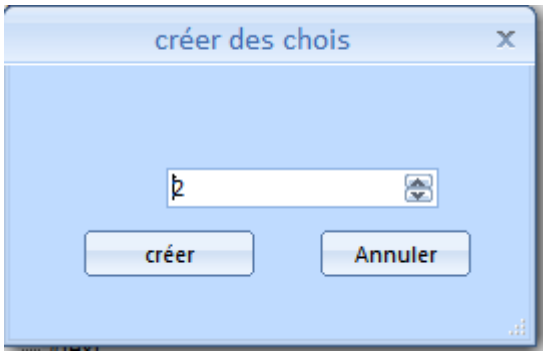


Figure. 4.6 fenêtre de choix pour une notion

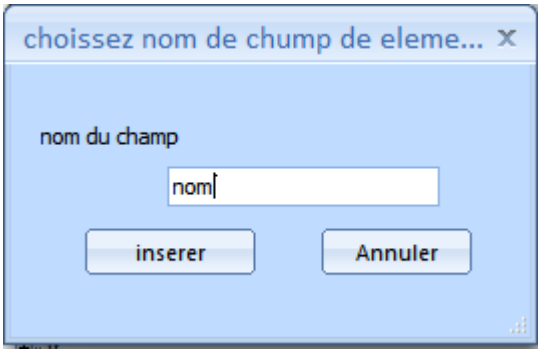


Figure. 4.7 fenêtre de metaelement

6- Apres avoir vu l'option (metaany,metavaluesAttributs :Photo :source, metavalue, metachoice). On prend le nœud table qui contient deux fils tr dans le cas telephone qui peuvent présenter plusieurs fois comme il peut ne pas être présent dans d'autre pages (0 ou N). Pour ce exemple on pointe vers le nœud table et crée metagroup (voir figure 4.8), on prend le nœud qui peut être regroupé (tel) en lui donne Min= 0 cas d'absence de téléphone dans une page et Max =N cas de présences de plusieurs téléphones, en laissant le nœud tr(Email) inchangé dans le nœud table qui sera modifié par un nœud metagroup qui contient un nœud fils group lui-même et qui porte la branche de la notion tel avec sa valeur. Ensuite on pointe vers le #texte <0556677798> en cliquant sur l'option metavalue et le nœud père <metaelement:Telephone>.

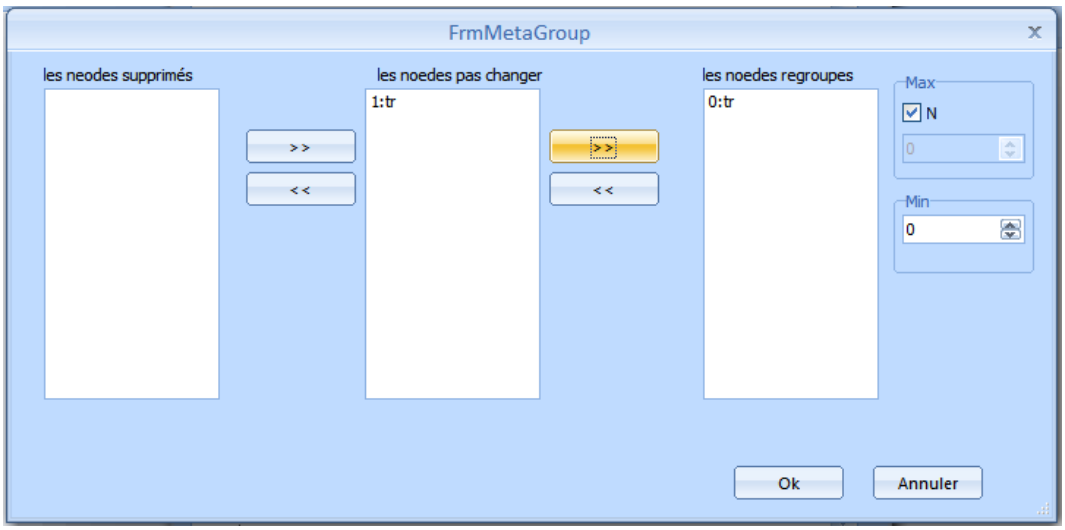


Figure. 4.8 Fenêtre MetaGroup

7- En continuant le même travail pour les autres metaelement avec Email.
8- On sauvegarde finalement le squelette.
9- En cas d'erreurs, le créateur de squelette permet d'annuler les actions qu'on a fait sur le squelette.

Apres une création de squelette on sauvegarde le projet pour pouvoir le recharger en cas d'annulations ou des modifications pour ne pas reprendre le travail dès le début. On passe ensuite à l'extraction se fait par le chargement de squelette et les pages html concernons à extraire on appuyant sur le boutant Démarrer à la fin en obtient un fichier résultat sous forme XML contient les données extraites.

Conclusion

L'extracteur qu'on a implémenté est un outil destiné aux utilisateurs particuliers et connaisseurs dans le monde de conception de site web qui est exploitable à 100% sans aucun problème s'il a un bon squelette solide sur ce côté on a consacré un bon temps pour réaliser le constructeur de squelette qui n'est pas à 100% mais on peut compter sur lui .

Conclusion générale

Dans ce mémoire nous avons présenté une démarche qui permet d'extraire les données à partir des pages html d'un site internet.

Cette démarche commence par une classification des pages html en groupes de pages semblables et de choisir de chaque groupe une page déléguée qui va être traitée.

Le traitement des pages déléguées est fait en deux phases : la première phase consiste à convertir la page html en xhtml à l'aide d'un outil de conversion prêt à l'emploi. La deuxième phase est la construction du squelette XML de la page. Le squelette est construit d'une manière semi-automatique (guidée par l'utilisateur), elle se résume par le choix de l'utilisateur des notions correctes auxquelles les données de la page html correspondent.

Une fois le squelette construit, on procède à l'extraction des données de toutes les pages du groupe et les enregistrer dans un fichier XMl, qui servira à la conception de la base de données.

Comme tout travail humain n'est jamais parfait, nous souhaitons que notre système soit complété avec d'autres fonctionnalités par ceux qui suivront ce travail.

Nous citons quelques idées pour compléter à merveille notre travail :

- Intégrer un module de classification automatique des pages html pour les regrouper en classes.
- Ajouter un module de passage de l'XML à la description de la structure de la table de la base de données.
- Travailler avec XLinq pour créer des relations entre les fichiers XML et donc entre les tables de la base de données.

Bibliographie

1. *Remy levy , ‘’Ré-ingénierie des données ‘’ Extrait de la lettre de l’ADELI N°28-1997.*
2. *Michael Hammer et James Champy ‘’LE REENGINEERING’’* Traduit de l'américain par Michel Le Seac'h dunod 1993.
3. *Linda Wills Philip Newcomb ‘’ REVERSE ENGINEERING ‘’ KLUWER ACADEMIC PUBLISHERS 1996.*
4. *Jean-Roch. Meurisse, ‘’ Extraction de données de site web, méthodologie, outils et étude de cas’’ Mémoire de licence en informatique faculté universitaire notre dame de la paix, Namur Institut d’Informatique, 2004.*
5. *Jean Marie-Favre, Jonathan-Musset , : ‘’Retro-Ingénierie’’ dirigée par les Meta modèles ;Secondes journées sur l’ingénierie dirigée par les modèles IDM 2006- Lille, France.*
6. *E. Chikofsky, J. Cross, ‘’Reverse Engineering and Design Recovery: A Taxonomy’’. IEEE Software, Jan. 1990.*
7. *R. Arnold, ‘’Software Reengineering’’, IEEE, ISBN 0-8186-3272-0, 1993.*
8. *Gérard Le blanc ‘’C# et .Net version2’’ ÉDITIONS EYROLLES 61, bd Saint-Germain 75240 Paris Cedex 05*
9. *Programmer dans .Net Framework avec le langage C# version 2.0_ 713 pages de cours + 94 pages d’exercices corrigés (2006).*
10. *FAQ XML Grégory Picavet - forum - GrandFather -Erwy (erwy.developpez.com) - Mathieu Lemoine (Mes articles pour Devloppez.com) – ‘’ Ont contribué à cette FAQ :XML ‘’ Date de publication : 20/04/2006.*
11. *FAQ (X)HTML Ont contribué à cette FAQ :Kerod - trotters213 – Master Of ChakhaL - Forum (X)HTML -Bisûnûrs - BrYs - Maxoo - Linaa - Giminik - Jérôme - debug- Florian - Manolo - Géronimo - necronick - Eric Berger -lunatix - prgasp77 - Yogui - mathieu - Yoshio - acmillenium -Date de publication : 15/09/2009.*
12. *Site de Html2Xhtml . ‘’http://www.it.uc3m.es/jaf/html2xhtml/’’*

ملخص

إعادة هيكلة المواقع تعتمد أساسا على التصميم العكسي للعناصر المكونة لها و المثيرة للاهتمام لا سيما تحويل مواقع ثابتة إلى مواقع حيوية مع إعادة بناء قاعدة بيانات الموقع ، في هذه المذكرة نقدم طريقة تسمح باستخراج البيانات من صفحات Html من موقع على شبكة الأنترنت من أجل استغلالها في إعادة الهيكلة و الهندسة العكسية لتصميم قاعدة البيانات .

كلمات مفتاحية : الهندسة العكسية، هيكل ، Xml، Xhtml، Html

Résumé

La restructuration des sites web et l'éventuelle rétro-conception de leurs éléments constitutifs peuvent s'avérer plus qu'intéressantes, notamment dans le cas de conversion des sites statiques en sites dynamiques ou de reconstruction de la base de données d'un site. Dans ce mémoire nous présentons une démarche qui permet d'extraire les données à partir des pages html d'un site internet en vue de les exploiter pour la reconstruction et la rétro-conception de la base de données.

Mot clés : Retro-ingénierie , Html, Xhtml, Xml, Squelette

Abstract

The restructuring of web sites and the possible reverse engineering of their components can be more interesting, especially in the case of transforming static sites into dynamic sites by the reconstruction of the database site. In this thesis we present an approach that can extract data from HTML pages of a website in order to exploit them for the reconstruction and reverse engineering of the database.

Key Words: Reverse-engineering, Html, Xhtml, Xml, Squelet