Rétro-ingénierie du modèle de présentation pour les pages Web

Laurent BOUILLON, Jean VANDERDONCKT

Université catholique de Louvain, Institut d'Administration et de Gestion,
Unité de Systèmes d'Information,
Belgian Lab. of Computer-Human Interaction (BCHI)
Place des Doyens, 1 – B-1348 Louvain-la-Neuve, Belgique
{bouillon, vanderdonckt}@isys.ucl.ac.be

Résumé : Cet article décrit un modèle, une méthode et un outil permettant d'effectuer une rétro-ingénierie de pages Web écrites en HTML, de manière à obtenir un ou plusieurs modèles de présentation. Ce modèle spécifie les différents objets de présentation contenus dans une page Web en les organisant en niveaux d'abstraction progressivement croissants : les objets d'interaction concrets (OIC), les objets d'interaction abstraits (OIA), les fenêtres logiques (FL) et les unités de présentation (UP). Différents modèles avec variantes peuvent être produits en fonction d'une série d'options et d'heuristiques de rétro-ingénierie portant sur les différents niveaux. A partir de ces modèles de présentation, une ingénierie progressive permet de produire une nouvelle IHM adaptée à un autre contexte d'utilisation.

Mots-clés : Rétro-ingénierie, migration, approche orientée modèle, modèle de présentation, analyse statique, page Web, technologie XML, heuristiques, XIML.

1 Introduction

La multiplication des plates-formes, et en particulier des plates-formes mobiles (tels que les téléphones mobiles, les ordinateurs de poches, etc.), a provoqué ces dernières années un accroissement considérable des contextes d'utilisation des applications. Par contexte d'utilisation, on entend une série de paramètres tels que le lieu, le système d'exploitation, les moyens d'interaction, le langage utilisé, etc.

Or, ces applications - conçues avant l'apparition des nouvelles plates-formes - ne proposent pas une interface utilisateur adaptée au contexte. Les travaux sur la plasticité de l'IHM (Thevenin, Coutaz, 1999), (Calvary, Coutaz, Thevenin, 2001) tentent de résoudre ce problème en proposant une méthode grâce à laquelle une interface est construite par sélection d'objet d'interaction selon les contraintes de la plate-forme ciblée et de l'environnement (moyens d'interaction, taille d'écran et de la fenêtre,…) à partir d'un modèle de tâche et concepts couvrant divers contextes.

Suite à l'incapacité de l'interface à prendre en compte les contextes d'utilisation différents, des applications interactives deviennent de plus en plus rapidement

obsolètes. Il existe donc un besoin réel d'adapter ces applications aux changements technologiques, de manière à en améliorer l'accessibilité ainsi que de proposer une interface homme-machine (IHM) ajustée au contexte d'utilisation. Les sites Web sont les premiers concernés, ceux-ci étant de plus en plus consultés depuis des navigateurs différents résidant sur des plates-formes différentes. Notre dépendance vis-à-vis de l'information provenant d'Internet grandira encore davantage dans le futur avec la variété des dispositifs d'accès à Internet (par exemple un Internet ScreenPhone, un SmartPhone), rendant ce besoin de plus en plus aigu.

La multiplication des plates-formes constitue un des facteurs encourageant la réingénierie d'applications dans la mesure où elle invite à déployer une même IHM pour plusieurs plates-formes simultanément. Il existe d'ailleurs un besoin constant dans les organisations de mettre à jour et de rénover les applications ayant atteint un âge critique pour diverses raisons : les besoins liés à l'activité changent, les tâches des utilisateurs changent, les utilisateurs eux-mêmes évoluent, l'infrastructure technologique est modernisée ou adaptée à des besoins nouveaux tels que le passage au nouveau millénaire (Vanderbrandt, Klint, Verhoef, 1996).

Le processus de la *réingénierie* (Chikofsky, Cross, 1990) d'IHM d'un contexte d'utilisation vers un autre peut être décomposé en deux étapes successives : *l'ingénierie régressive* (ou rétro-ingénierie) qui consiste à analyser, à parcelliser une application de manière à en extraire une représentation abstraite (indépendante de la plate-forme et du langage) et *l'ingénierie progressive* durant laquelle une nouvelle interface est générée à partir des spécifications actualisées et d'une représentation abstraite. L'utilité d'une telle méthode se révèle dans la possibilité de produire plusieurs interfaces pour plusieurs contextes d'utilisation et donc pour différentes plates-formes (un cas particulier parmi les divers contextes d'utilisation possibles) à partir des spécifications.

En ce qui concerne le problème particulier de la rétro-ingénierie de pages Web écrites en HTML en vue de leur utilisation dans différents contextes, tels que différentes plates-formes, plusieurs candidats s'offrent à nous pour convertir du HTML vers un autre langage de balisage : W4F (Sahuguet, Azavant, 1999), les transformations XSL (Clark, 1999) ou les conversions du HTML au XHTML (Pemberton, 2000).

Ces alternatives ont été rejetées pour les raisons suivantes : les transformations supportées sont essentiellement d'ordre syntaxique, ne sont pas paramétrables, sont limitées à certaines opérations de base. Ces transformations ne permettent pas de supporter une rétro-ingénierie voulue comme flexible et procédant à des sélections complexes en fonction des contextes. Ceci nous a amené à développer une application dédiée au problème rencontré : VAQUITA (pour reVerse engineering of Applications by Questions, Information selection and Transformation Alternatives).

L'approche proposée ici permet d'effectuer une rétro-ingénierie de pages HTML de manière à en obtenir un modèle de présentation, utilisable par la suite en ingénierie progressive afin de recomposer une nouvelle interface pour un autre contexte d'utilisation. Ce modèle de présentation est exprimé en un langage qui se veut indépendant de celui utilisé dans les contextes différents. Cette approche serait surtout utile pour les développeurs de sites Web désirant améliorer l'accessibilité de leurs pages sans recommencer entièrement la conception. L'analyse des pages ne porte pour le moment que sur les objets codés dans le langage de balisage HTML 4.0, c'est-à-dire sans prise en compte des langages de script (tels que Perl ou JavaScript) ni des applets Java.

Cet article est organisé comme suit : la deuxième section contient un état de l'art de différents travaux relatifs à la rétro-ingénierie des IHM pour d'autres contextes, la troisième section définit les niveaux d'abstraction du modèle de présentation qui est le résultat de la rétro-ingénierie proposée ici, la quatrième section définit les concepts fondamentaux du XIML (le langage de spécification dans lequel le modèle de présentation est spécifié), la cinquième section décrit l'architecture de l'application VAQUITA, la sixième partie détaille une étude de cas et finalement la septième section est dédiée à la conclusion et aux travaux futurs.

2 Etat de l'art en rétro-ingénierie et migration d'IHM

Avant d'expliciter l'approche de VAQUITA, un bref aperçu d'autres techniques de rétro-ingénierie est donné dans cette section. Ces techniques passent toutes par une représentation intermédiaire abstraite avant de recomposer une nouvelle IHM.

2.1 AUIDL

L'environnement AUIDL (Elwahidi, Merlo, 1995) reste le pionnier en matière de rétro-ingénierie d'interface. Cet environnement possède son langage associé IDL (Interface Development Language). L'interface textuelle est d'abord traduite dans les modèles de présentation et du dialogue écrit en IDL. Ensuite, ces spécifications sont converties dans le langage EASEL, permettant la génération automatique des écrans d'une IHM pour l'environnement IBM 3270. Les spécifications EASEL couvrent le modèle de présentation (composé de structures hiérarchiques d'objets interactifs concrets) et le modèle du dialogue (décrivant le comportement de ces OIC - Objets d'Interaction Concrets). Ce nouveau code généré est finalement lié à l'ancien noyau sémantique.

2.2 Remplacement d'une interface DOS par une interface Windows

Cette approche (Csaba, 1997) procède par le contrôle de l'affichage d'une application écrite pour DOS par l'intermédiaire d'une autre application afin d'effectuer une rétro-ingénierie de son IHM devenue obsolète. La méthode utilisée est de type dynamique, se basant sur une reconnaissance des informations et événements envoyés et reçus par l'application originale. Une deuxième application utilise ces informations afin de les replacer dans les objets appropriés de l'interface Windows. Le programme lie ensuite la nouvelle interface avec le restant de l'application DOS grâce à une couche intermédiaire.

2.3 MORPH

MORPH (Moore, Rugaber, 1997), (Moore, Rugaber, Seaver, 1994) est un processus de re-génération d'interface utilisateur textuelle vers une interface utilisateur graphique supporté par un outil. Le processus de MORPH (Model Oriented Reengineering process for Human-Computer Interface) se décompose en trois étapes :

1) la détection des tâches d'interaction : Morph identifie les tâches d'interactions basiques par une analyse statique du code, incluant l'analyse de contrôle des flux (« control flow analysis »), l'analyse des flux de données et de mises en correspondance. Une analyse dynamique peut être introduite afin de résoudre les cas qui ne peuvent être résolus statiquement.

2) la représentation qui est la recomposition d'un modèle abstrait de l'interface grâce aux tâches précédemment détectées. Les informations recueillies lors de l'étape précédente sont parcourues afin de déterminer les entrées/sorties réalisée par l'ancienne application en les identifiant comme

objets de données (« data objects »). Ces objets permettent de lier la nouvelle interface et l'ancien noyau fonctionnel grâce à une classification de ces objets selon leur type de données (sélection dans une liste, entrée/sortie quantifiée, entrée/sortie texte, positionnement) et leur localisation précise dans le code.

3) la transformation de ce modèle vers une nouvelle interface graphique. Cette étape de transformation est conduite grâce à une base de connaissances qui permet de sélectionner les différents objets concrets selon les attributs abstraits détectés. Cette base de connaissance regroupe une série de règles permettant d'améliorer les capacités d'inférence pour produire une nouvelle IHM. Finalement, Morph permet de générer des interfaces pour OSF/Motif, Java ou Ms-Windows.

2.4 CELlest

Le projet CELlest (Stroulia *et al.*, 2000), (Kong *et al.*, 99), (Stroulia, Kapoor, 2002) est l'acronyme de CEL Legacy Enhancement Software Technologies. Cette approche propose une nouvelle manière de procéder en rétro-ingénierie qui consiste à comprendre les processus logiques de l'ancienne application par l'analyse de son utilisation au travers d'un modèle de la tâche. Cette rétro-ingénierie est donc de type dynamique. Il comprend l'outil URGENT (User interface ReGENeration Tool) qui produit ce résultat en trois étapes :

1) l'analyse de la tâche : un enregistreur stocke les transitions entre les écrans et les actions de base nécessaires pour réaliser une tâche déterminée (analyse de l'activité pour réaliser une tâche comme « établir une facture d'assurance »), pour plusieurs occurrences pour un même utilisateur ou différents utilisateurs, en cours de session interactive ;

2) la spécification abstraite de l'IHM : de ces données capturées lors de la première étape est déduite une IHM abstraite adaptée à la tâche effective par abstraction (par exemple l'utilisateur ne doit saisir une donnée qu'une seule fois) et simplification (par exemple l'utilisateur ne doit pas changer d'écran). L'abstraction consiste à effacer les actions redondantes, comme entrer deux fois son login pour deux applications différentes. La simplification revient à afficher des parties de plusieurs écrans provenant de (ou des) l'ancienne(s) application(s) sur un même écran dans l'interface finale ;

3) la génération de la nouvelle IHM : à partir de cette spécification, une IHM en DHTML est dynamiquement générée avec sélection des OIC associés à chaque donnée.

L'avantage de l'approche de CELlest réside en une génération d'une interface directement adaptée à une tâche effective plutôt que prévue dans la mesure où elle a été estimée par analyse des traces d'interaction. Cette méthode vise la génération d'une seule interface pour un objectif déterminé et peut donc faire intervenir les noyaux fonctionnels de plusieurs anciennes applications afin de réaliser une tâche interactive particulière.

2.5 WARE

WARE (Web Applications Reverse Engineering) (Di Lucca *et al.*, 2001, 2002) se base essentiellement sur les aspects dynamiques de la rétro-ingénierie (par exemple, les objets statiques tels que le texte, les tables, … ne sont pas analysés) et vise à faciliter la maintenance et l'évolution des applications Web par la création d'une base de connaissances contenant les informations prélevées lors de la rétro-

ingénierie. Les résultats sont stockés sous la forme de diagrammes notés en UML. Ce processus se divise en trois analyses :

1) Statique : durant cette analyse, les différents composants et leurs relations sont détectés et classifiés dans un diagramme de classes UML. Une nouvelle classe est créée pour chaque type d'élément HTML différent, les liens sont traduits sous la forme de relations et les paramètres sont transformés soit en attributs, soit en nouvelles classes.

2) Dynamique : cette étape consiste à repérer les différents liens entre objets d'interactions ainsi que la localisation de ceux-ci dans le code. Durant cette étape, les classes trouvées en rétro-ingénierie statique sont validées ou complétées. A partir de ces informations, il est possible de générer les diagrammes de séquence et de collaboration utiles à la nouvelle IHM.

3) Comportementale : cette partie de la rétro-ingénierie vise la modélisation du comportement fonctionnel grâce à l'analyse des dialogues de collaborations et séquences et par l'étude de l'utilisation de l'application. Le résultat de cette étape est la constitution de diagrammes de cas d'utilisations à partir du comportement fonctionnel.

Cette rétro-ingénierie est réalisée uniquement pour générer une représentation abstraite de la structure des objets (surtout leurs relations) appartenant à une application Web et de cette manière, à en faciliter la maintenance. Le tableau 1 synthétise les différents processus de rétro-ingénierie évoqués dans cette section.

	AUIDL	**Csaba**	**Morph**	**Cellest (Urgent)**	**WARE**
Type d'IHM en entrée	Textuelle	DOS (texte)	Textuelle	Textuelle	Graphique (applications HTML)
Méthode d'analyse	Statique	Dynamique	Statique et Dynamique	Dynamique	Statique, Dynamique et Comportementale
Langage abstrait	IDL	Spécifique	Spécifique	Spécifique	UML
Modèles	Présentation, Dialogue	/	Présentation, Tâche	Présentation, Domaine, Tâche	Class, Collaboration / Sequence, Use Case
Type d'IHM en sortie	Ms-Windows	Ms-Windows	Java, OSF/Motif, Ms-Windows	HTML, XHTML, WML, Java	/

Tableau 1 – *Synthèse des propriétés des processus de migration*

En particulier, la ligne *Modèles* relate les différents types de modèles couverts par la rétro-ingénierie. Parmi ceux-ci, le modèle de la présentation occupe une place de choix tant par la capacité à retrouver un tel modèle à partir des informations disponibles que par l'importance fondamentale que revêt ce modèle dans l'ensemble du cycle de vie. La section suivante s'attache à décrire plus spécifiquement le modèle de présentation qui est le résultat du processus de rétro-ingénierie décrit dans cet article.

3 Le modèle de présentation

Une IHM est composée d'une série d'objets interactifs avec lesquels l'utilisateur peut interagir. Ces objets sont répertoriés et hiérarchisés dans le modèle de présentation, en prélevant toutes les informations relatives à leur apparence.

3.1 Les concepts théoriques

Le modèle de présentation (Bouillon, Vanderdonckt, Eisenstein, 2002 ; Barclay, Kennedy, 2000 ; Szekely, 1996) a été mis en place afin de pallier plusieurs manquements du HTML. Tout d'abord, la structure du code HTML ne reflète pas une décomposition de la présentation en termes d'objets d'interaction. En effet, certains attributs en HTML sont écrits sous la forme de balises et sont mis sur un même pied d'égalité syntaxique qu'une table par exemple. Par exemple, la balise <b> fait passer le style des libellés contenus dans cette balise en gras. Si l'on écrit en HTML une phrase avec un seul mot en gras, celui-ci sera placé sur un niveau hiérarchique plus bas que le restant de la phrase, ce qui ne correspond pas à une hiérarchie en objets d'interaction. Rien n'a été prévu non plus dans ce langage pour effectuer des regroupements sémantiques aux niveaux supérieurs. Or, ces regroupements sont indispensables pour procéder à une ingénierie progressive. C'est donc principalement pour élaborer une structure abstraite du code et permettre l'ingénierie progressive qu'intervient notre modèle de présentation.

Le modèle de présentation est la représentation abstraite des objets d'interaction concrets appartenant à une IHM (fig. 1). Un *objet interactif concret* (OIC) est un objet réel appartenant à une interface utilisateur déterminée qu'un utilisateur peut voir ou manipuler comme, par exemple, une boîte à cocher.

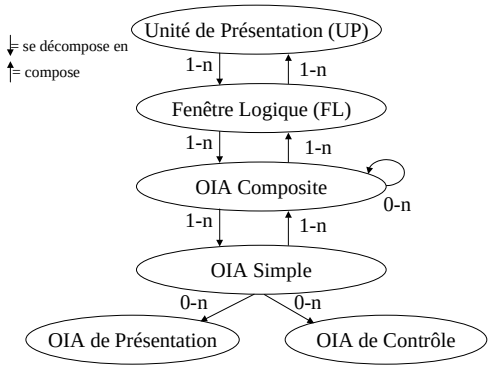


Figure 1 – *Hiérarchie des concepts de présentation*

Un objet d'interaction est dit *simple* lorsqu'il ne peut être décomposé en d'autres OIC plus petits. Il est dit *composite* dans le cas contraire. Si une interaction est possible avec l'OIC, il est appelé OIC de *contrôle*. Dans le cas opposé, c'est un OIC de *présentation*. Un OIC de présentation est purement statique et ne sert qu'à favoriser une meilleure présentation dans le format, la structuration. Par exemple, une boîte de regroupement, un séparateur. Un *objet d'interaction abstrait* (OIA) est une abstraction des OIC du point de vue de leur présentation et de leur comportement, de manière à en avoir une représentation indépendante de toute contrainte logicielle et matérielle. Chaque OIA est identifié par un nom générique unique (par exemple : un libellé) associé à ses attributs abstraits (généraux et particuliers à l'objet). A

chaque OIA peut correspondre zéro, un ou plusieurs OIC ; ce nombre dépend de l'OIA et de la plate-forme sur laquelle on veut concrétiser l'OIC. Par exemple, un champ d'édition abstrait peut être concrétisé dans les environnements Windows par un « single-line entry field », Mac par un « text entry field » et Open Look par un « Single line entry field ».

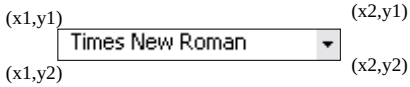
Chaque OIA (Vanderdonckt, Bodart, 1993) possède une multitude d'attributs abstraits définissant ses caractéristiques visuelles. Lors de la transformation de l'OIA en OIC sur une plate-forme spécifique, une sélection selon ses attributs doit être faite entre les différents OIC disponibles correspondant à l'objet abstrait, mais cette étape appartient déjà à l'ingénierie progressive.

On distingue encore deux abstractions dans le modèle de présentation : la fenêtre logique et l'unité de présentation. La *fenêtre logique* (FL) est un OIA composite car elle contient différents OIA simples ou composites. La fenêtre logique peut être considérée soit comme la fenêtre physique ou une boîte de dialogue, soit comme un conteneur logique d'autres OIA.

L'unité de présentation aborde la question du lien avec la tâche. L'*unité de présentation* (UP) est sensée contenir tout l'environnement de présentation nécessaire à l'exécution d'une tâche déterminée. Chaque unité de présentation peut contenir (au moins) une à plusieurs fenêtres logiques, affichées simultanément ou en alternance à l'écran. La fenêtre de départ de l'unité de présentation est appelée *fenêtre de base* à partir de laquelle il est possible de naviguer vers d'autres fenêtres logiques.

3.2 Les relations de placement

Les relations de placement déterminent le positionnement des OIA dans une unité de présentation ou une fenêtre logique. Dans notre approche, ces relations de placements sont exprimées de manière relative. Ce type de placement est à opposer aux placements absolus pour lesquels une référence est faite à un système de mesure, comme le pixel. Les possibilités de placements absolus en HTML sont de plusieurs types, dont voici quelques définitions :



- La succession (a,b) : l'objet a est situé sur la droite de l'objet b ⇔ $x2_a > x1_b$
- L'infériorité (a,b) : l'objet a est situé en dessous de l'objet b ⇔ $y2_a > y1_b$
- La composition (a,b) : l'objet a est contenu dans l'objet b ⇔ $x1_a > x1_b$ et $x2_a < x2_b$ et $y1_a > y1_b$ et $y2_a < y2_b$
- La composition centrée : l'objet a est contenu dans l'objet b et a est centré par rapport à b ⇔ composition (a,b) et $x2_a - x1_a = x2_b - x1_b$
- La justification gauche (a,b) ou (b,a) : l'objet a est justifié à gauche sur l'objet b ou inversement (relation symétrique) ⇔ $((y2_a > y1_b)$ ou $(y2_b > y1_a))$ et $x1_a = x1_b$

Ces relations de placement sont enregistrées en XIML (cf. section 4) sous la forme de « relation_statement », qui est un type particulier d'attribut d'un objet d'interaction. Cette relation est indiquée comme un attribut d'un objet source (a dans les exemples ci-dessus) faisant référence à un autre objet (b dans les exemples) et possède également un type (infériorité, composition, …).

Les différentes relations employées pour modéliser le placement des objets en HTML sont classables en deux catégories -dont au moins une de chacune des deux classes est présente dans le placement de chaque objet- , les descriptions de relations horizontales et verticales. Ces deux types sont obligatoires pour pouvoir déterminer

la position absolue des objets d'interaction concrets lors de l'ingénierie progressive. Toutefois, il est possible qu'un élément possède plusieurs relations verticales ou horizontales. Par exemple, l'image possédera deux relations de succession (fig. 2), en l'occurrence succession (libellé1) et succession (libellé2).

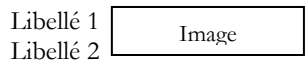


Figure 2 – *Plusieurs relations horizontales pour un même objet*

Ces deux relations auraient pu également être modélisées par la précédence (l'inverse de la succession), de manière à n'utiliser qu'une seule relation de chaque type (horizontal et verticale), mais celle-ci n'est pas employée dans notre approche. En effet, celle-ci demande de faire référence à un objet non encore détecté dans le flux de l'analyse et il est donc plus aisé d'implémenter une solution faisant référence aux objets déjà détectés. Le même raisonnement est appliqué à l'infériorité par rapport à la supériorité.

3.3 La démarche

La plupart des pages HTML contiennent les OIA « classiques », tels que des boîtes à cocher, listes de sélection déroulantes, etc., que l'on peut retrouver dans d'autres types d'interface utilisateur. Toutefois, une série d'objets HTML sont propres au domaine d'Internet et ont dû être rajoutés aux OIA existants (comme les liens hypertextes, les cartes cliquables, …). D'autres OIA existant ont dû faire l'objet de modifications ou d'expansions (par exemple : les tables, boutons de commande, etc.) dans leur définition de certains attributs ou dans les valeurs permises par ceux-ci.

Les objets d'interaction sont tous traduits en XIML sous la forme d'éléments de présentation dans le modèle de présentation, qu'ils soient simples ou composites, une fenêtre logique ou une unité de présentation. C'est la position hiérarchique de l'élément dans le modèle qui détermine à quel type de concept appartient l'élément. La section suivante est consacrée au langage XIML utilisé lors de la rétro-ingénierie pour stocker le modèle de la présentation des pages Web.

4 Le langage de modélisation d'interface

XIML (Puerta, Eisenstein, 2001) est l'abréviation de eXtensible user-Interface Markup Language (http://www.ximl.org). Ce langage, basé sur XML, fournit un ensemble de spécifications servant à décrire une interface utilisateur en termes abstraits, c'est-à-dire en se voulant le plus indépendant possible de toute implémentation de cette interface sur une plate-forme donnée. Ces spécifications se veulent indépendantes de toute contingence logicielle et matérielle.

Le principe du XIML est basé sur une approche de construction de l'interface utilisateur orientée modèles. Les modèles repris dans un fichier XIML sont les modèles de tâche, de la présentation, du dialogue, de l'utilisateur, du domaine, de la plate-forme, des préférences et du modèle 'à usage général'. Ce dernier peut être défini par l'utilisateur afin de tenir compte d'informations supplémentaires dans certains cas particuliers.

Deux caractéristiques importantes du XIML sont l'extensibilité et la flexibilité. En effet, il existe la possibilité de rajouter de nouvelles balises sans problème de compatibilité. Lors de l'apparition de nouvelles plates-formes ou technologies, les anciens fichiers XIML peuvent être aisément modifiés afin que les nouveaux

modèles tiennent compte des changements. Ce langage est également flexible car il est ouvert : toute personne a la possibilité de modifier les définitions du langage original afin de mieux décrire l'interface utilisateur qu'elle désire générer sans pour autant changer les concepts fondamentaux de sa spécification. Le XIML permet également une mise à jour des composants de l'IHM. Les spécifications du XIML contiennent les adresses Internet des objets d'interactions concrets de l'interface utilisateur. Comme l'interface peut être compilée à l'exécution, lorsqu'il y a eu des changements apportés par un concepteur à un de ses composants, ceux-ci peuvent être instantanément pris en compte.

Les déclarations de spécifications XIML se font à l'aide de quatre concepts fondamentaux (Puerta, Eisenstein, 2002) :

- **Modèle** : on distingue deux types de modèle, le modèle d'interface et les composants du modèle. Le premier est la racine de tout document XIML et contient les différents sous-modèles disponibles en XIML. Tous ne doivent pas être présents simultanément et un même type de modèle-composant peut exister plusieurs fois sous le même modèle d'interface. Les composants du modèle (tâche, domaine, utilisateur, présentation, dialogue, plate-forme, préférences et le modèle général) contiennent toutes les informations spécifiques à une dimension de l'interface.
- **Elément** : représente toute information décrivant un modèle-composant. Dans le cas des éléments de présentation, c'est une unité d'information décrivant l'apparence visuelle d'une interface utilisateur. Chaque élément de présentation peut en contenir d'autres (jusqu'aux AIO simples, c'est-à-dire indécomposables). Par exemple une fenêtre est un élément de présentation pouvant contenir l'élément de présentation 'table' qui contient lui-même d'autres éléments, etc. Un élément de présentation peut faire référence à un objet extérieur au code, par exemple un contrôle ActiveX, une image ou une police de caractères. Dans ce cas, l'attribut « Location » peut être utilisé afin de spécifier une URL où l'objet pourra être trouvé.
- **Attribut** : représente une simple unité d'information déclarative à propos d'un modèle d'interface, d'un modèle-composant ou d'un élément. Il faut toujours définir un attribut pour ensuite pouvoir le déclarer dans une instance particulière. Cette définition de l'attribut est composée de la liste de ses valeurs permises, de la valeur par défaut de l'attribut, de sa forme canonique, de la documentation (information sur ce qu'il représente) et de son type.
- **Relation** : définit un lien entre les éléments et/ou les modèles. L'élément auquel se réfère le lien est spécifié grâce à l'attribut « reference », contenant l'identificateur auquel l'objet se rapporte. Toute relation doit être définie préalablement et cette définition contient les classes permises dans la relation ainsi que son identificateur.

Une portion de site Web (fig. 3) demandant de remplir un champ d'édition unilinéaire devant contenir l'e-mail de l'utilisateur est traduite à l'aide de ces quatre concepts de la manière suivante :

```
<Interface ID= « _01 »>
<Modele_de_présentation id= «demande_email»
    <Elément_de_présentation id= « fenêtre »>
    <Nom> fenêtre principale </nom>
          < Elément_de_présentation id = « EmailTextBox »>
          <Nom>Champ d'édition de l'E-mail</nom>
          <Caractéristiques>
  B       <Etablissement_attribut id= « Size »>25</Etablissement_attribut>
          <Etablissement_attribut id= « MaxSize »>45 </Etablissement_attribut >
          </Caractéristiques >
          </Presentation_element>
A         <Presentation_element id= « AddresseEmailLabel »>
          <Nom>Libellé indiquant la signification du champ d'édition</nom>
          <Caractéristiques>
          <Etablissement_attribut id= « LabelColor »>#000000
          </ Etablissement_attribut>
          <Etablissement_attribut id= « LabelStyle »>Normal
      C   </Etablissement_attribut>
          <Etablissement_attribut id= « LabelString»>Adresse E-mail :
          </ Etablissement_attribut >
          …
          <Etablissement_relation Definition= « Label_describes »
          reference= « EmailTextBox »>
          </Elément_de_presentation>
</Elément_de_presentation>
</Modèle_de_presentation>
```

Figure 3 – *Exemple de spécifications XIML*

Le modèle d'interface inclut le modèle de présentation contenant l'élément de présentation 'fenêtre'. Cette fenêtre contient l'élément de présentation 'champ d'édition' ainsi que l'élément de présentation 'libellé' décrivant la saisie d'information. La taille à l'écran ainsi que la taille maximale de la chaîne de caractères que le champ d'édition peut contenir, sont définies dans ses attributs. Le libellé possède également plusieurs attributs, comme la taille de caractère, la couleur, etc. Comme ce libellé décrit le champ d'édition, grâce à la relation « Label_describes », il est possible de définir le lien sémantique entre ces deux objets. Le XIML enregistre ensuite ces informations de manière déclarative.

5 L'application de rétro-ingénierie

5.1 Caractéristiques de la méthode utilisée

La méthode de rétro-ingénierie utilisée (Bouillon, Vanderdonckt, Souchon, 2002) est de mode statique, c'est-à-dire avant l'exécution du code. Le mode statique est à l'opposé de la rétro-ingénierie dynamique qui consiste à modifier la page tout en l'exécutant. L'avantage de l'analyse statique est une meilleure qualité de l'approche, si des contraintes locales à un niveau plus élevé que le code (cf. options) doivent être introduites, ce qui est le cas pour VAQUITA (http://www.isys.ucl.ac.be/bchi/research/vaquita.htm). L'analyse dynamique se prête plus à la modélisation de la tâche, de l'utilisateur ou du dialogue dans lesquels les interactions ont plus

d'importance. La méthode introduit également la variabilité des pages sujettes à rétro-ingénierie. Une rétro-ingénierie peut représenter exactement l'interface qu'elle transforme (consistance), ou éliminer certains éléments et en modifier d'autres afin de rendre la page plus utilisable, mieux adaptée aux contraintes de la nouvelle plate-forme (variable).

Une autre dimension de la rétro-ingénierie concerne la proportion des possibilités et tâches traduites. On peut soit viser la rétro-ingénierie totale de la page (toutes les sous-tâches proposées par la première interface), soit ne transformer que les sous-tâches utilisables par la nouvelle interface. En effet, selon les contraintes locales et les capacités d'interaction, il est parfois impossible de préserver le même degré d'utilisabilité ou toutes les sous-tâches de l'interface d'origine. La méthode utilisée ici tente de factoriser les différentes parties de l'interface selon la caractéristique qu'elles puissent être utilisées selon plusieurs contextes d'utilisation.

En résumé, la méthode de rétro-ingénierie est de type statique et tente de maintenir un maximum de consistance dans la présentation et dans les possibilités offertes par l'interface originale tout en tenant compte des contraintes locales. L'introduction d'options permet de moduler la rétro-ingénierie vers le type d'interface que l'utilisateur souhaite obtenir. La section suivante décrit le processus automatisé pour la rétro-ingénierie du code HTML.

5.2 L'application

VAQUITA est l'application développée en MS-Visual Basic afin de rétro-ingénier les pages HTML. La fenêtre principale de l'application est représentée à la fig. 4 (Vanderdonckt, Bouillon, Souchon, 2001).

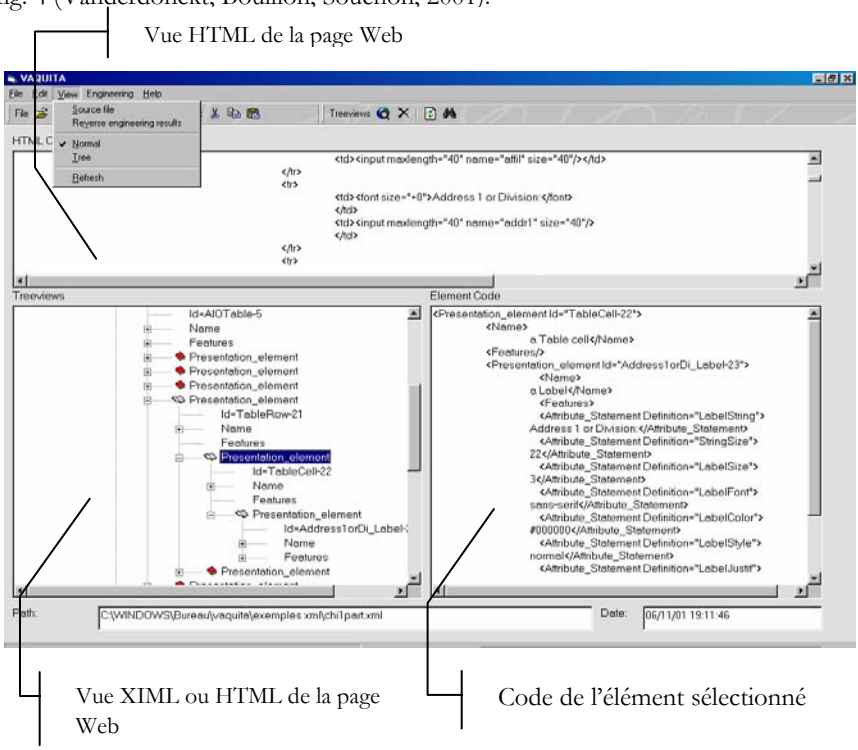


Figure 4 – *Les trois vues de VAQUITA*

Celle-ci est divisée en trois zones : un champ d'édition dans le haut de la fenêtre contient le code HTML à analyser, une structure arborescente sur la gauche de la fenêtre affiche la structure du document analysé ainsi que celle du modèle de présentation (résultats), et un dernier champ d'édition contient les spécifications – en HTML ou en XIML. Lors de la sélection d'un élément XIML dans la vue en structure arborescente, les spécifications associées à l'élément de présentation sont affichées dans la vue dédiée au code. Dans un même temps, la portion de code HTML correspondante à l'élément sélectionné est affichée dans la « vue HTML de la page Web ». Ces trois parties sont justifiées par le besoin d'une visualisation de l'information dite du « foyer et du contexte » (Furnas, 1986).

Une autre visualisation des deux codes est également disponible (fig. 5), permettant de consulter simultanément le code HTML et XIML affichés sous forme de structure arborescentes. Cette vue permet de comparer les structures de code et de vérifier la traduction du HTML en objets abstraits selon les besoins de l'utilisateur.

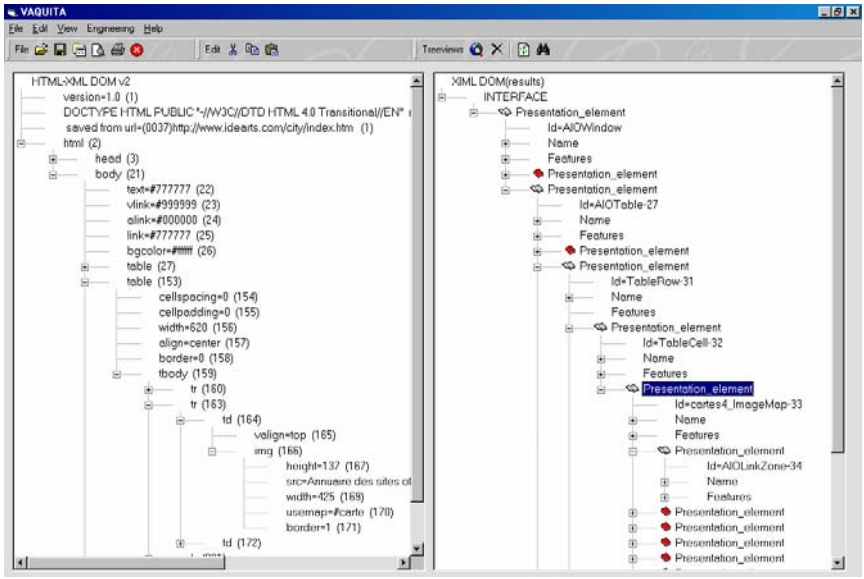


Figure 5 – *Deuxième interface de VAQUITA*

L'avantage de VAQUITA est la possibilité d'obtenir plusieurs modèles de présentation à partir d'une même page par la sélection d'heuristiques de rétro-ingénierie différentes. Ceci permet à l'utilisateur d'explorer diverses rétro-ingénierie et d'observer les conséquences des heuristiques et options sélectionnées sur le modèle produit. Ces choix se font à partir de la page des options de rétro-ingénierie (fig. 6).

Les options de rétro-ingénierie sont divisibles en deux types : celles se rapportant aux OIA et leurs attributs, et celles concernant les heuristiques.

1. Il est possible de sélectionner les objets à analyser par sélection des boîtes à cocher correspondantes. Il est alors possible d'ignorer la détection de certains OIA ou de déterminer, objet par objet, quels attributs sont à prendre en compte lors de la rétro-ingénierie. On peut ainsi supporter virtuellement n'importe quelle rétro-ingénierie de n'importe quelle balise HTML et de ses

éléments par omission ou inclusion de ceux-ci. Ceci est particulièrement pratique pour restreindre une rétro-ingénierie uniquement aux OIC sujets à interaction, sans prendre en compte les enjolivements graphiques tels que les bannières, les publicités, les logos.

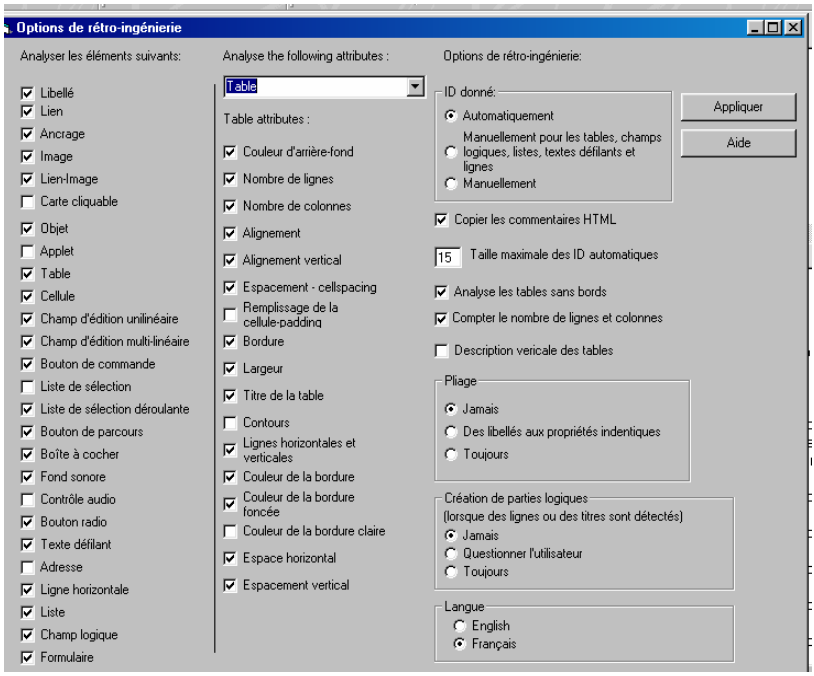


Figure 6 – *Page des options de VAQUITA*

2. Comme il est possible d'effectuer la rétro-ingénierie d'une page selon différents critères et règles de traduction, le choix des techniques à utiliser est laissé au développeur. Le deuxième type d'option permet d'utiliser diverses techniques de traduction de la page. Ces options comportent notamment :

- le pliage (« folding ») qui consiste à fusionner plusieurs libellés consécutifs dont on ignore s'ils font partie d'un même texte sémantique ou non. Ceci peut s'appliquer soit aux libellés ayant des propriétés identiques (strict) telles que les mêmes attributs de présentation (par exemple même police de caractère, même taille, même casse, même couleur), soit à tous les libellés (non-strict). Inversement, l'option de dépliage permet de traduire une telle série comme une succession de plusieurs OIA 'libellé' avec une déclaration de relation entre eux afin de préserver leur séquence.
- L'option 'Ne pas analyser les tables ne possédant pas de bordure' permet de supprimer les tables non-affichées à l'écran. Une manière courante en HTML de s'assurer la bonne structuration d'une page lors de l'affichage est de déclarer les éléments dans des tables sans bords. Celles-ci peuvent être ignorées en XIML et remplacées par la déclaration de relations de placement équivalentes.
- Couverture de cellules (« cell span ») : l'utilisateur décide si les cellules couvrant plusieurs colonnes doivent être traduites comme un seul

élément de présentation ou comme une succession de plusieurs éléments de présentation si la disposition n'est pas considérée comme importante.

- Traduction des tables : cette option permet d'effectuer la rétro-ingénierie des tables par balayage horizontal ou vertical par la génération d'un élément de présentation soit pour chaque ligne soit pour chaque colonne. Chaque ligne ou colonne est ensuite décomposée en une série d'éléments de présentation représentant les cellules de tables.

L'application est disponible en deux langues, le français et l'anglais, alternables grâce à un bouton radio sur cette même page d'options.

5.3 L'architecture

L'ensemble des traitements réalisés lors de la rétro-ingénierie d'une page Web est représenté sur la fig. 7.

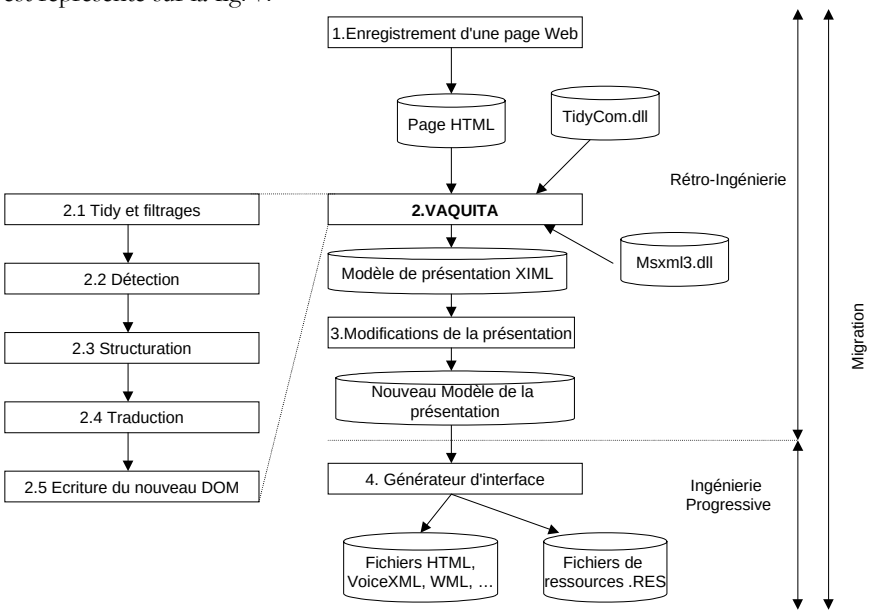


Figure 7 – *Traitements réalisés pour la migration d'une page HTML*

1) Enregistrement d'une page Web : le processus démarre par la sauvegarde locale d'une page Web car VAQUITA ne peut traiter les pages directement sur Internet.

2) VAQUITA : Le fichier **msxml3.dll** (accessible sur http://www.msdn.microsoft.com/xml/articles/mscabfile.asp) fournit une série de fonctions servant aux traitements de documents écrits en langage à balises de type XML, ce qui a facilité l'implémentation de VAQUITA. Il est alors possible de placer le document HTML dans une structure DOM (Document Object Model), et d'effectuer des recherches par éléments, d'en créer de nouveaux, de les modifier, etc.

Grâce au fichier partagé msxml3, il est possible de naviguer de nœud en nœud par une procédure récursive et de repérer les balises correspondant à des objets interactifs concrets dans l'ordre de leur apparition. L'ordre d'apparition dans le code est important car l'affichage des différents éléments se fait dans cet ordre, du fait que ceux-ci n'ont pas de coordonnées. Lors de la conversion en XIML, il est toutefois possible de déclarer de manière assistée certaines relations (par exemple un

libellé est rattaché à un champ d'édition) entre les éléments de présentation, ce qui permet de mieux rendre compte de la présentation désirée de la page. Quant à la bibliothèque **tidycom.dll** (téléchargeable sur http://perso. wanadoo.fr/ablavier/ tidyCom/), elle fournit toutes les fonctions offertes par le logiciel gratuit Tidy. Cette application corrige les éventuelles erreurs de syntaxe et les balises superposées, structure le code d'une manière plus claire et efface les balises inutiles. Le traitement effectué par VAQUITA est encore décomposable en cinq sous-traitements :

2.1) L'étape « Tidy et filtrages » consiste en une correction automatique de la syntaxe du code HTML lorsque le code n'est pas syntaxiquement bien formé. De nombreux navigateurs sont très permissifs au niveau de la syntaxe et corrigent généralement ce genre d'erreur à l'interprétation. Ce traitement permet de ne devoir traiter qu'un seul type de page, car Tidy permet également de transformer la présentation du code. Le grand avantage de l'utilisation de la bibliothèque TidyCom est la possibilité de transformer les fichiers HTML en XML valide. Aux traitements réalisés par Tidy ont été rajoutées des procédures de filtrage pour les pages écrites en français. En effet, les accents ne sont pas des caractères XML valides et sont donc remplacés dans le code (par exemple « é » devient « e/aigu »). Une fois la page transformée, il est possible d'employer une API DOM pour traiter plus aisément la page.

2.2) La détection se fait d'abord par classification d'éléments. Lorsqu'un élément est rencontré, une variable reçoit une valeur correspondant à l'élément. Ensuite intervient une recherche des attributs ciblés selon la valeur de la variable afin de ne pas parcourir toutes les possibilités de classification à chaque itération de la boucle de détection.

2.3) La structuration du fichier des résultats se fait selon les règles de traduction de la hiérarchie du modèle de présentation. Selon les options, l'utilisateur peut intervenir à différentes étapes de ce processus afin de produire le document structuré de la manière désirée.

2.4) La traduction des balises en OIA et attributs est réalisée grâce à l'utilisation de tables de traduction. A titre d'exemple, une table de traduction (tableau 2) d'un élément et de ses attributs est reprise ci-dessous. La colonne de gauche contient la représentation en XIML de la balise HTML 'textarea'. Dans ce cas, les valeurs des attributs en HTML sont reprises dans les valeur des attributs XIML sans modifications. Si le champ d'édition multilinéaire n'est pas accessible (par l'ajout de l'attribut 'disabled' dans la balise 'textarea'), alors l'attribut EDMEnabled prend la valeur faux.

2.5) l'écriture du nouveau Document Object Model est la dernière étape au cours de laquelle les éléments traduits sont écrits dans une autre structure DOM pour deux raisons : d'abord pour écrire un nouveau document XML contenant le modèle de présentation grâce à de nombreuses méthodes (fonctions) d' édition fournies par la bibliothèque msxml (telles que la création ou le remplacement de nœuds du DOM) et ensuite pour faciliter son affichage dans une structure arborescente (les fonctions DOM et de structure arborescente sont pratiquement identiques) qui permettra à l'utilisateur de visualiser plus facilement le modèle de présentation.

Champ d'édition multilinéaire (extended edit box)	**textarea**
ID	name=+*EDM*
EDMNumberOfLines	rows=
EDMNumberOfCols	cols=
EDMEnabled = faux	disabled
EDMDefaultString	value=
EDMAccessKey	Accesskey=
EDMTabIndex	TabIndex=
EDMReadOnly	ReadOnly=

Tableau 2 – *Table de traduction du HTML vers le XIML*

3) Modifications de la présentation : les informations contenues dans le fichier HTML une fois enregistrées sous le format XIML peuvent encore faire l'objet de modifications afin de transformer l'apparence de la page ou pour la rendre plus adaptée à certaines plates-formes. Les modifications peuvent porter aussi bien sur les éléments de présentation que sur leurs attributs. Il est par exemple possible d'ajouter de nouveaux éléments permettant de mieux structurer la page, de modifier l'emplacement de certains éléments par simple utilisation de la souris ('drag & drop'), d'opérer des retraits d'éléments ou de modifier la valeur de certains attributs. Une modification possible serait de transformer tous les attributs définissant des couleurs d'une page en couleurs noir et blanc pour avoir une interface accessible aux plates-formes monochromes. Un éditeur de code XIML existe déjà à cet effet, permettant les modifications citées ci-dessus (fig. 8).

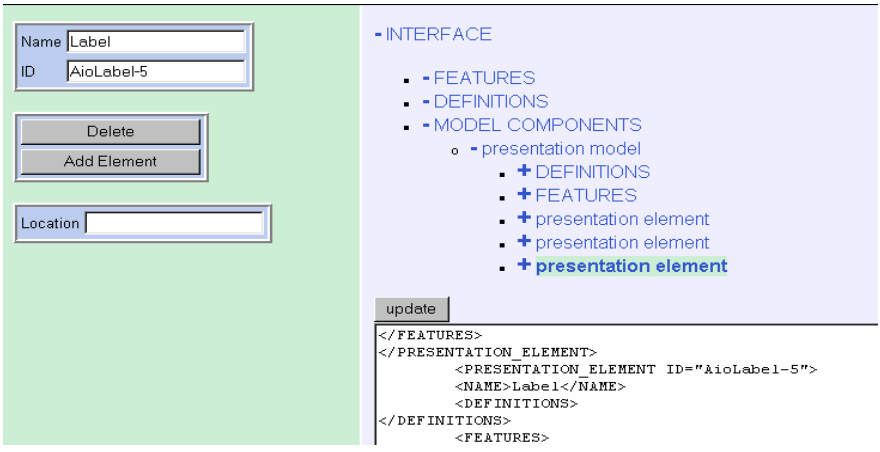


Figure 8 – *Editeur XIML*

4) Générateur d'interface : après avoir effectué la rétro-ingénierie de la page, il est possible d'utiliser une autre application pour la reconversion des spécifications abstraites en code spécifique à une plate-forme. Cet outil assure la conversion vers le HTML, le WML, le XML et Java. La rétro-ingénierie présentée ici peut être utilisée en boucle pour la reconception d'une page. En effet, une page peut être stockée dans un modèle de présentation, qui peut à son tour être modifié puis utilisé pour la génération d'une nouvelle page. Cette boucle pourrait améliorer l'utilisabilité

d'une page Web en considérant d'autres aspects, comme le modèle du dialogue, de l'utilisateur et de la plate-forme. Sur les figures ci-dessous (fig. 9a, 9b et 9c) sont reprises quelques illustrations de potentielles conversions du XIML vers différents langages d'une petite portion du formulaire.

Figure 9a – *Représentation sur un téléphone portable*

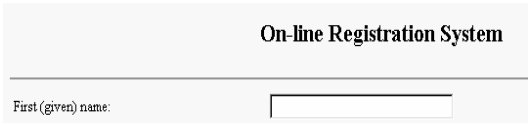


Figure 9b – *Représentation sur un terminal*

Figure 9c – *Représentation sur une interface VB*

6 Etude de cas

6.1 Rétro-ingénierie d'une page

La page Internet (fig. 10) servant à cette étude de cas est la page d'inscription en ligne pour la conférence CHI'2001 accessible sur http://www.acm.org/sigchi/chi2001/registration.html. Cette page a été choisie car elle peut être considérée comme relativement représentative et réaliste d'une page HTML que l'on peut trouver sur un site Web. Celle-ci contient un formulaire ('form') d'inscription, composé de nombreux libellés, champs d'éditions, boîte à cocher, bouton radio, tableaux à cellules variables, images, etc. Tout d'abord, une déclaration de l'interface a été écrite manuellement en XIML afin d'avoir un point de comparaison avec les résultats obtenus par l'application et pour valider les résultats obtenus automatiquement par rapport à ceux obtenus manuellement.

Cette page peut être traduite suivant les concepts du modèle de présentation de la manière suivante : la page entière constitue l'unité de présentation (UP), le concept le plus élevé hiérarchiquement du modèle. Cette UP contient quatre fenêtres logiques (les quatre points principaux) correspondant aux divisions sémantiques de la page. La première FL renferme le titre de la page, la seconde est dédiée aux informations personnelles (nom, adresse, courrier électronique, etc.) de l'utilisateur. La troisième fenêtre contient quelques requêtes spéciales concernant l'inscription à la conférence et enfin la dernière division concerne l'inscription à la conférence (jours, sous quel statut, etc.).

La division en FL est arbitraire et cette page pourrait faire l'objet d'autres subdivisions comme par exemple fractionner la deuxième FL en deux autres FL, l'une se rapportant au nom et à l'affiliation de l'utilisateur, et l'autre se référant à des informations moins importantes.

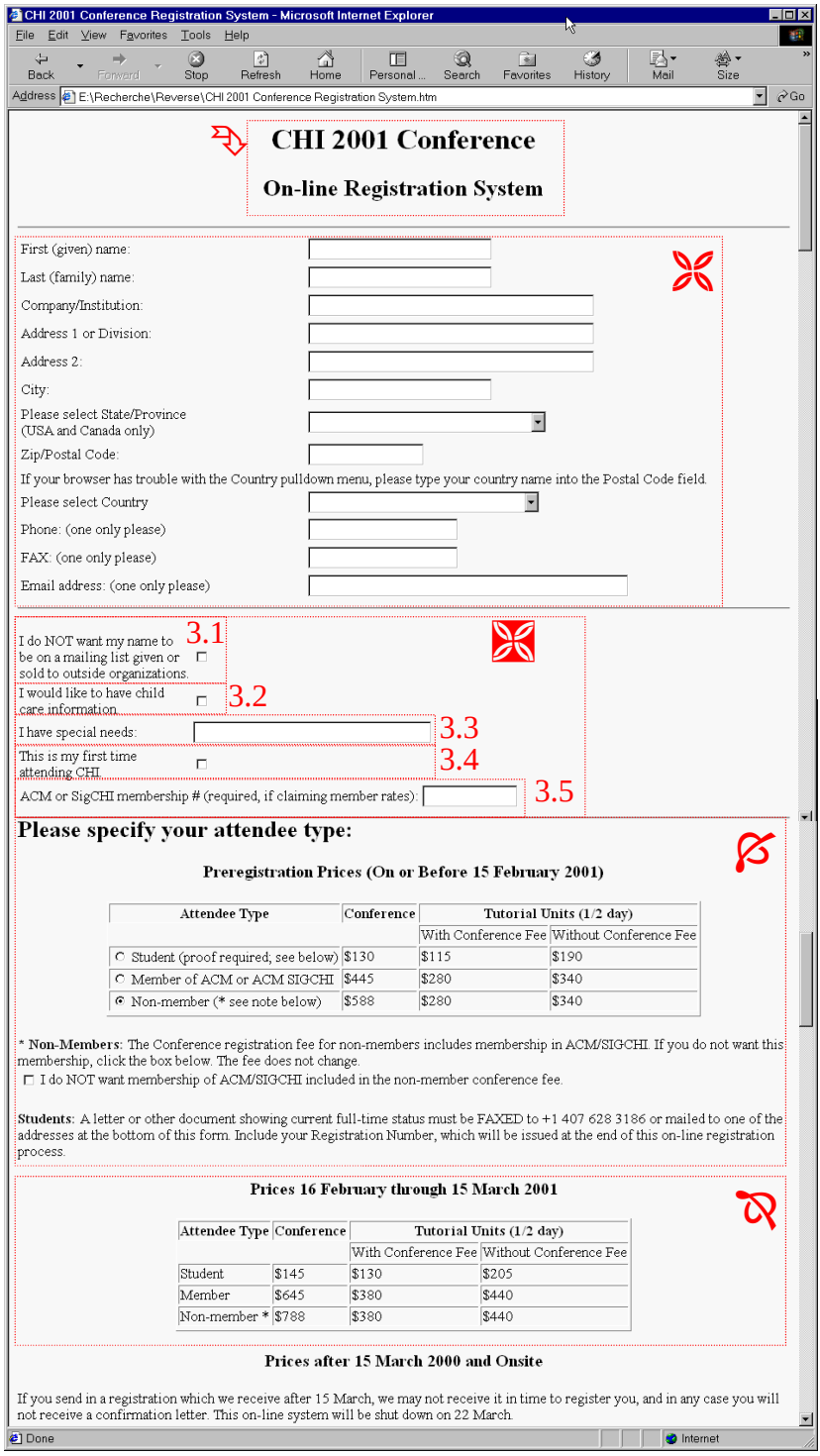


Figure 10 – *Enregistrement en ligne à la conférence CHI'2001*

Chaque FL contient plusieurs OIA composites (OIAC) ou simples (OIAS). Par exemple, la section 3 contient cinq objets d'interaction composites. Les objets 3.1, 3.2 et 3.4 sont trois objets composés de deux OIA simples : un libellé et une boîte à cocher. De même, les objets 3.3 et 3.5 sont deux OIA composites contenant deux OIA simples, un libellé et un champ d'édition. Le champ d'édition permet une interaction avec l'utilisateur et est de ce fait classifié comme OIA de contrôle au contraire du libellé qui est un OIA de présentation. Un autre exemple d'OIA composite serait le tableau de la quatrième fenêtre logique, contenant une série de cellules elles même composées d'OIA simples.

Les objets d'interaction concrets sont tous traduits dans le modèle de présentation comme éléments de présentation, selon une hiérarchie déterminant leur classification (fig. 11). Cette figure explicite complètement le modèle pour trois objets (3.1, 3.2 et 3.3 sur la fig. 10) et ne reprend pas la structure entière du modèle pour cette page Web.

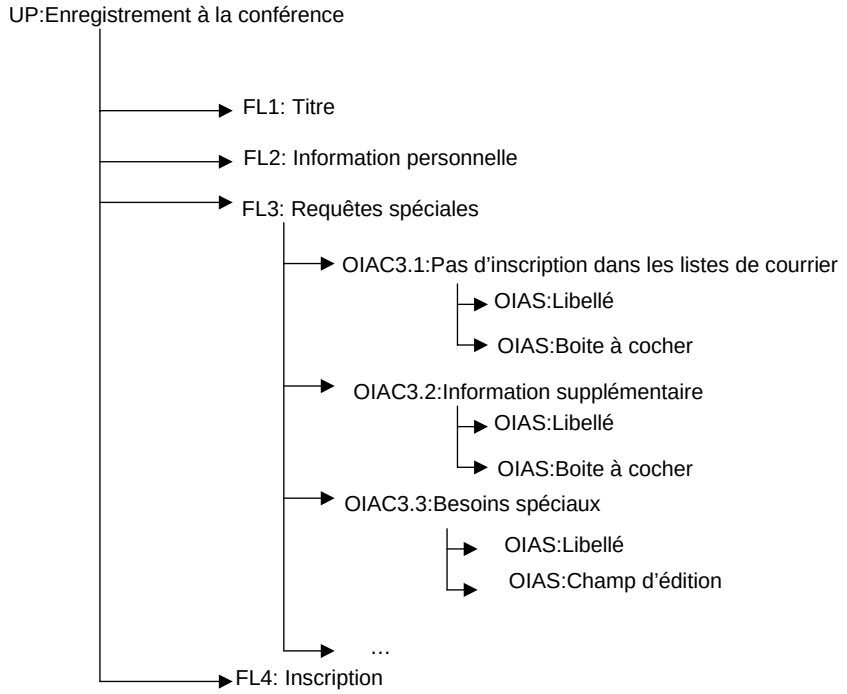


Figure 11 – *Structure du modèle de présentation*

La portion du modèle de présentation (ci-dessus) est ensuite traduite en XIML :

```
<Elément_de_présentation ID=«Enregistrement_Conférence»>
   <Elément_de_présentation ID=«Titre»>
   <Elément_de_présentation ID=«Infos_Personnelles»>
   <Elément_de_présentation ID=«Requêtes_spéciales»>
      <Elément_de_présentation ID=«Pas_liste_courrier»>
          <Elément_de_présentation ID=«AIOLibellé»>
          <Elément_de_présentation ID=«AIOBoite_à_cocher»>
      <Elément_de_présentation ID=«Infos_supplémentaire»>
          <Elément_de_présentation ID=«AIOLibellé»>
```

```
        <Elément_de_présentation ID=«AIOBoite_à_cocher»>
    <Elément_de_présentation ID=«Besoins_spéciaux»>
        <Elément_de_présentation ID=«AIOLibellé»>
        <Elément_de_présentation ID=«AIOChamp_d'édition»>
<Elément_de_présentation ID=«Inscription»>
```

Les résultats obtenus lors de la rétro-ingénierie sont repris dans le tableau 3. Celles-ci ont été obtenues sur Pentium-Celeron 433 Mhz possédant 128 Mb de RAM.

Caractéristiques	CHI Conference registration HTML	CHI Conference registration XIML
Taille (ko)	7,95	59,9
Nombre de caractères	7524	57910
Nombre de lignes	316	2215
Balises et attributs	493	1655
OIA (éléments de présentation)	-	156
Temps d'une analyse (secondes)	-	126
Temps d'une analyse sans la détection de certaines balises (*)	-	98
Temps d'une analyse « accélérée ».	-	95
Erreurs		0 %

Tableau 3 – *Performances lors de l'étude de cas*

Le taux d'erreur en ce qui concerne la hiérarchie du modèle de présentation et la traduction des balises en éléments de présentation est nul pour cette étude de cas. Ceci est essentiellement dû au fait que cette page n'a pas été créée à l'aide d'une application de création de page Web (par exemple Frontpage ou Dreamweaver) qui ajoutent une série de balises inutiles. Ces balises créent parfois des erreurs lors de l'analyse du type d'objet.

L'analyse accélérée est une rétro-ingénierie pour laquelle la détection de certains attributs a été supprimée car ceux-ci demandent le parcours entier de la page, et ce parfois à plusieurs reprises, alors que ces attributs pourraient être considérés comme secondaires. Par exemple, la détermination du nombre de colonnes des tables nécessite, dans le cas de tables imbriquées, plusieurs parcours de grandes portions de code.

L'analyse pour laquelle la détection de certaines balises a été supprimée (Cf. (*) dans le tableau 3) sert de témoin pour l'analyse accélérée (ces balises sont les objets, applets, fonds sonores, contrôles audio, cellules, textes défilant, adresses et les champs logiques). Cela montre que le fait d'ignorer quelques attributs diminue plus le temps de traitement que la suppression de huit éléments entiers, mais peu fréquents pour la plupart.

La différence de taille des spécifications XIML (presque de 60 Ko) par rapport à la page HTML (d'environ 8 Ko) est due au fait que les spécifications XIML sont extrêmement nombreuses, précises et détaillées. Un exemple de cette augmentation de précision sont les instructions HTML donnant certaines caractéristiques aux éléments qu'elles englobent ('basefont', 'tr', 'center', …). Ces instructions ne sont plus reprises comme telles dans le code XIML, mais remplacées dans les attributs des éléments compris entre ces instructions HTML, et ce pour chaque élément

(multiplication du nombre d'attributs). Cela permet la suppression ou le déplacement de ces éléments sans affecter la présentation d'origine.

Un exemple d'un type de rétro-ingénierie possible de l'étude de cas est repris ci-dessous, pour une portion de la page CHI2001 Registration (fig. 12). Cette portion est composée de cinq objets interactifs abstraits contenus dans une fenêtre. Le code de chaque objet est commenté par la suite.

Figure 12 – *Portion de la page d'inscription à la conférence CHI*

```
(A)<Modèle_de_présentation ID="CHIRegPres">
<Nom>Presentation model for CHI Conference registration
 </Nom>
<Elément_de_présentation ID="CHIRegWindow">
<Nom>CHI registration window</Nom>
<Caractéristiques>
 <Etablissement_attribut DEFINITION="AIOWindowTitle">
 CHI 2001 Conference Registration System
 </Etablissement_attribut>
 <Etablissement_attribut DEFINITION="AIOWindowBgColor">
 #f8f8f8</Etablissement_attribut>
</Caractéristiques>

(B)<Elément_de_présentation ID="CHIregTitle">
    <Nom>CHI conference registration title</Nom>
    <Elément_de_présentation ID="RegIntroLabel">
    <Nom>Registration Intro Title</Nom>
    <Caractéristiques>
      <Etablissement_attribut DEFINITION="LabelString">
      CHI 2001 Conference </Etablissement_attribut>
      <Etablissement_attribut DEFINITION="LabelLevel">
      1</Etablissement_attribut>
      <Etablissement_attribut DEFINITION="LabelJustif">
      middle</Etablissement_attribut>
      </Caractéristiques>
   </Elément_de_présentation>
   <Elément_de_présentation ID="OLRegSysLabel">
    <Nom>On Line Registration System Title</Nom>
    <Caractéristiques>
    <Etablissement_attribut DEFINITION="LabelString">
     On-line Registration System </Etablissement_attribut>
    <Etablissement_attribut DEFINITION="LabelJustif">
     middle</Etablissement_attribut>
    <Etablissement_attribut DEFINITION="LabelLevel">
     2</ATTRIBUTE_STATEMENT>
    </Caractéristiques>
```

```
   </Elément_de_présentation>
   <Elément_de_présentation ID="IntroHrule">
    <Nom>End of Intro Horizontal rule</Nom>
    <Caractéristiques>
    …
    </Caractéristiques>
   </Elément_de_présentation>
</Elément_de_présentation>

(C)<Elément_de_présentation ID="PersonnalInfo">
    <Nom>Personnal info</Nom>
    <Elément_de_présentation ID="FirstNaTextBox">
     <Nom>First name TextBox</Nom>
     <Caractéristiques>
      <Etablissement_attribut DEFINITION="EDXSize">
       25</ATTRIBUTE_STATEMENT>
      <Etablissement_attribut DEFINITION="EDXMaxSize">
       25</Etablissement_attribut>
     </Caractéristiques>

    <Etablissement_attribut ID="FirstNameLabel">
     <Nom>First Name </Nom>
       <Caractéristiques>
        <Etablissement_relation DEFINITION="Label_describes"
         REFERENCE="FirstNameTextBox">
        </Etablissement_relation>
        <Etablissement_attribut DEFINITION="LabelString">
         First (given) name:</Etablissement_attribut>
       </Caractéristiques>
    </Elément_de_présentation>
  </Elément_de_présentation >
```

(A) Le premier élément du code XIML est la déclaration du modèle de présentation contenant la première fenêtre logique, qui est également la fenêtre physique de la page. L'élément est identifié de manière unique par l'attribut ID, et c'est par celui-ci que l'on peut encore faire référence à l'élément dans la suite du code. La balise 'Nom' contient une description plus précise sur l'objet et n'a pas d'autre valeur que de donner plus d'information sur l'élément. Les attributs des objets sont définis par défaut en XIML, et si les attributs d'un OIC sont différents de ceux de base, ceux-ci sont précisés dans la balise 'Caractéristiques'. Dans ce cas-ci, la fenêtre possède deux attributs propres, le titre et la couleur de fond.

(B) Le premier élément est une fenêtre logique de la page, à savoir l'en-tête du formulaire d'inscription. Comme ceci est une division sémantique, cet élément ne possède pas d'attributs, mais englobe trois éléments, à savoir le titre, le sous-titre et la ligne horizontale. Si l'option 'folding non-strict' avait été sélectionnée, il n'y aurait eu qu'un seul libellé, car ils auraient été fusionnés. Ces libellés possèdent tous deux les attributs se référant à leur taille de caractères, de leur alignement et de la chaîne de caractère les composant.

(C) À nouveau, le premier élément est une déclaration d'une fenêtre logique de la page. Celle-ci englobe tous les libellés et saisie d'informations relatives aux informations personnelles de l'utilisateur. Celle-ci englobe donc une série de champs d'édition et de libellés, contenus dans un tableau. Dans cette rétro-ingénierie, l'option d'ignorer les tableaux non-affichés a été sélectionnée. En conséquence, le deuxième élément de cette partie est un champ d'édition servant à contenir le prénom de l'utilisateur et celui-ci n'est donc pas contenu dans une cellule de table. Ses deux attributs représentent la taille du champ affiché à l'écran (en caractères) et la taille maximale de la chaîne qu'il peut contenir. L'élément suivant est le libellé précisant le contenu du champ d'édition. Comme ces deux éléments sont reliés sémantiquement, le libellé fait l'objet d'une déclaration de relation vers le champ d'édition correspondant. Le nom de la relation-type est « label_describes » et l'objet auquel le champ d'édition se réfère est « FirstNameTextBox ». La position du libellé par rapport au champ est « à gauche (left) », mais comme c'est la position par défaut, elle n'a pas besoin d'être précisée.

En conclusion de l'étude de cas, la traduction automatique du code a donné des résultats en tous points identiques avec l'analyse préalable de la page en ce qui concerne les éléments détectés, leur ordre d'apparition et leur hiérarchie. La seule différence résidait dans la déclaration des relations de libellés aux autres objets, qui est inexistante dans le procédé automatisé. Ces relations permettent de spécifier dans les attributs d'un libellé l'objet que le texte décrit. Ce dernier point pose plusieurs problèmes lors de l'implémentation du procédé automatique, car les relations ne peuvent être créées au moment de la détection du libellé. En effet, l'élément auquel celui-ci se réfère peut ne pas avoir encore été détecté. Cette étape doit donc se faire après l'analyse complète de la page, en re-parcourant les éléments créés. Une autre difficulté pour ce type de relation est l'incertitude des déductions. Comme souligné dans (Sussman, Banavar, Bergman, 2001), certains cas peuvent posséder une ambiguïté qui ne peut être résolue de manière automatisée. Par exemple, pour le cas d'une boîte à cocher entourée de deux libellés, il est impossible - sans intervention humaine - de déterminer avec certitude le libellé descriptif de cet OIC.

Ces relations sont néanmoins déclarables grâce à une boîte de dialogue spécifique, mais cette opération reste rébarbative, même si aucune saisie au clavier n'est nécessaire. Toutefois, il est possible de cette manière d'obtenir exactement les mêmes résultats que lors de l'écriture manuelle de l'étude de cas. L'automatisation de ce processus est d'ailleurs en cours de développement, de manière à ne proposer une sélection des libellés les plus probables (proximité physique) pouvant être liés avec un objet d'interaction.

6.2 Performances

D'autres pages HTML ont encore été analysées afin de tester la capacité de traduction de toutes les balises HTML 4.0 par VAQUITA ainsi que les performances sur d'autres types de structure de page. Ces analyses (tableau 4) portent sur deux autres pages : « l'annuaire des sites officiels de tourisme en Belgique » qui est composé d'une carte cliquable, de diverses images cliquables et de très peu de libellés. La page d'« introduction au tutorial python» ne présente pratiquement que du texte et quelques tableaux.

On peut constater que le taux d'erreurs pour ces deux pages est d'environ 2 %, car celles-ci ont sans doute été générées à partir d'un logiciel d'aide à la conception de pages Web. L'analyse de plusieurs pages amène la constatation suivante : la taille

du fichier n'a pas d'influence sur le temps nécessaire à la rétro-ingénierie, au contraire du nombre de lignes. En effet, la taille des deux fichiers est presque identique à celle de l'étude de cas pour des temps d'analyse bien inférieurs. Le nombre de lignes correspond plus ou moins au nombre de balises en HTML et est généralement proportionnel au nombre d'éléments de présentation en XIML, ce qui est déterminant pour le temps nécessaire à la rétro-ingénierie.

Caractéristiques	Annuaire des sites de tourisme en HTML	Annuaire des sites de tourisme en XIML	Tutorial Python en HTML	Tutorial Python en XIML
Taille(ko)	6,29	43,7	7,68	44,5
Nombre de caractères	6693	42374	8671	43162
Nombre de lignes	128	1575	162	1339
Balises et attributs	349	1091	290	1225
OIA	-	75	-	76
Temps d'une analyse (secondes)	-	74	-	83
Temps d'une analyse sans la détection de certaines balises (cf. table 1)	-	64	-	73
Temps d'une analyse « accélérée ».	-	56	-	65
Taux d'erreurs		2 %		2 %

Tableau 4 – *Performances de VAQUITA*

7 Conclusion et travaux futurs

L'objectif principal de l'application décrite dans cet article est la rétro-ingénierie de pages HTML afin d'obtenir la génération automatique d'un modèle de présentation en XIML. Cette génération se veut automatique, mais hautement flexible grâce au jeu des nombreuses options de rétro-ingénierie en vue d'explorer des rétro-conceptions alternatives. Cet objectif s'inscrit dans un cadre bien plus large, qui est celui de l'adaptation d'anciens systèmes aux changements dans l'environnement (modifications dues à diverses raisons, telles que l'apparition de nouvelles technologies ou des changements organisationnels), la ré-ingénierie. Les changements à effectuer actuellement sont réalisés dans le but d'améliorer l'accessibilité des applications conçues anciennement pour un groupe de plates-formes spécifiques. Le but final consiste à déclarer l'interface d'une application en un langage neutre décrivant les sept modèles de base en XIML, de façon à rendre possible la recomposition de cette interface lors de l'exécution pour n'importe quelle plate-forme, selon ses caractéristiques et selon le contexte d'utilisation.

L'approche utilisée pour la rétro-ingénierie est donc l'approche par les modèles. Celle-ci permet de diviser le problème en différentes facettes et ce pour deux raisons : d'abord afin de déclarer l'interface dans un langage hautement spécialisé et précis, et ensuite pour réduire la complexité du problème en appliquant une décomposition du problème en sous-problèmes. Lors de l'étape ultérieure,

l'ingénierie progressive, les modèles sont recomposés afin d'obtenir un nouveau code.

Le langage HTML est un langage orienté vers la présentation d'items, le code d'une page ne contient généralement que des zones d'affichage ou de saisies d'informations. C'est donc pratiquement la page entière qui est analysée. La rétro-ingénierie se fait par analyse statique (la page n'est pas exécutée) et par des règles de traduction directes vers le XIML ou indirectes. Ces règles de traduction peuvent être totalement utilisées « sur mesure », jusqu'aux attributs des éléments de présentation, de façon à ce que l'utilisateur obtienne la rétro-ingénierie désirée. C'est l'avantage principal de l'application, car les autres outils de rétro-ingénierie disponibles actuellement ne proposent pas cette manière de procéder. Grâce à ces différentes options, le développeur est libre d'explorer différentes configurations de celles-ci afin d'obtenir le modèle de présentation le plus adéquat pour le cas qu'il désire traiter (reconception de la page, migration vers d'autres plates-formes, …). Ces options peuvent être enregistrées dans un fichier de configuration réutilisable de manière cohérente d'une rétro-ingénierie à l'autre.

Néanmoins, VAQUITA comporte à ce stade de nombreux manques, qui feront l'objet de travaux ultérieurs :

- Le processus présenté ici n'effectue pas la rétro-ingénierie pour les objets HTML non standards tels que les scripts en JavaScript, les scripts CGI, etc., ainsi que les feuilles de styles (suffixées .css). L'analyse des scripts en particulier permettrait de considérer les comportements dynamiques des objets (comme les changements d'apparence des boutons lorsque la souris est positionnée sur l'un d'eux), de manière à obtenir un modèle de présentation plus complet.
- L'approche ne prend pas en compte la navigation dans un même site. Il serait possible de modéliser le dialogue par une analyse de la navigation. Cette analyse peut être tant statique (par exemple par détection des sources de liens) que dynamique (par exemple par examen des liens parcourus à l'exécution par l'utilisateur).
- L'approche n'effectue la rétro-ingénierie que d'une page Web à la fois : l'identification d'informations effectuée au sein d'une page ne peut donc être réutilisée dans une autre page. La rétro-ingénierie du modèle du domaine a déjà fait l'objet de nombreux travaux (Hainaut *et al.*, 1995 ; Edwards, Munro, 1995) et s'avère utile lorsque l'on peut reconstituer un tel modèle par examen de plusieurs pages différentes contenant chacune des fragments du modèle du domaine. Il pourrait en être de même pour le modèle considéré ici.
- La rétro-ingénierie est limitée à une analyse statique. Grâce à une analyse dynamique, il serait possible d'améliorer la reconception d'une page pour un autre contexte d'utilisation, en prenant en compte la manière dont l'utilisateur interagit réellement avec une page Web.

Enfin, la rétro-ingénierie prise isolément pour elle-même n'a pas de sens. Qu'elle soit suivie par une ingénierie progressive, telle est la suite du processus pour aboutir à une IHM finale, exécutable dans un autre contexte d'utilisation. La disponibilité de générateur de code à partir d'une spécification rédigée en XIML autorise cette seconde étape. Une fois un modèle de présentation obtenu en XIML, il est possible de re-générer une autre IHM en HTML (mais il devrait être possible au moins de « retrouver » la même IHM de départ), en WML ou en un autre langage

à partir du moment où il est supporté par un outil approprié. Le champ d'investigation est donc large. Gageons qu'il puisse s'assortir de modèles, de méthodes et d'outils incorporant davantage l'expérience acquise par le concepteur en matière de rétro-ingénierie au fur et à mesure de ses pérégrinations.

Page web de référence

Une page web de référence décrivant les outils et travaux en cours dans le domaine de la rétro-ingénierie des IHM est accessible à http://www.isys.ucl.ac.be/bchi/research/soare.htm. Toute information relative à cet état de l'art peut être communiquée à l'adresse bouillon@isys.ucl.ac.be en vue de maintenir la page à jour.

Copyright

Références bibliographiques

Barclay, P.J. & Kennedy, J. (2000). Teallach's Presentation Model, in *Proc. of the 5th ACM Int. Working Conf. on Advanced Visual Interfaces AVI'2002* (Palermo, 23-26 mai 2000), ACM Press, New York, 2000, 151–154.

Bouillon, L., Vanderdonckt, J., & Souchon, N. (2002). Recovering Alternative Presentation Models of a Web Page with VAQUITA. Chapter 27, in C. Kolski & J. Vanderdonckt (eds.), *Proc. of 4th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2002* (Valenciennes, 15-17 mai 2002), Kluwer Academics Pub., Dordrecht, 2002, 311–322.

Bouillon, L., Vanderdonckt, J., & Eisenstein, J. (2002). Model-Based Approaches to Reengineering Web Pages, in *Proc. of 1st Int. Workshop on Task Model and Diagrams for User Interface Design TAMODIA'2002* (Bucarest, 18-19 juillet 2002), Academy of Economic Studies of Bucharest, Bucharest, 2002, 86–94.

Calvary, G., Coutaz, J., & Thevenin, D. (2001). Supporting Context Changes for Plastic User Interfaces: Process and Mechanism, in *Joint Proceedings of HCI'2001 and IHM'2001, People and Computers XV-Interaction without Frontiers* (Lille, 10-14 septembre 2001), A. Blandford, J. Vanderdonckt, & Ph. Gray (éds.), Springer-Verlag, Londres, 2001, 349–364.

Chikofsky E.J. & Cross, J.H. (janvier 1990). Reverse Engineering and Design Recovery: A Taxonomy, *IEEE Software*, Vol. 1, No. 7, 13–17.

Clark, J. (16 novembre 1999). XSL Transformations (XSLT) Version 1.0, W3C Recommendation, World Wide Web Consortium, accessible à http://www.w3.org/TR/xslt

Csaba, L. (1997). Experience with User Interface Reengineering – Transferring DOS panels to Windows, in *Proc. of 1st Euromicro Working Conference on Software Maintenance and Reengineering CSMR'97* (Berlin, 17-19 mars 1997), IEEE Computer Society Press, Los Alamitos, 150–156.

Di Lucca, G.A., Di Penta, M. , Antoniol, G., & Casazza, G. (2001). An approach for Reverse Engineering of Web-Based Applications, in *Proc. of 8th IEEE Working Conference on Reverse Engineering WCRE'2001* (Stuttgart, 5-7 octobre 2001), IEEE Computer Society Press, Los Alamitos, 231–240.

Di Lucca, G.A., Fasolino, A.R., Pace, F., Tramontana, P., & De Carlini, U. (2002). WARE: a Tool for the Reverse Engineering of Web Applications, in *Proc. of 6th European Conference on Software Maintenance and Reengineering CSMR'2002* (Budapest, 11-13 mars 2002), IEEE Computer Society Press, Los Alamitos, 241–250.

Edwards, H.M. & Munro, M. (1995). Deriving a Logical Data Model for a System Using the RECAST Method, in *Proceedings of the 2nd Working Conference on Reverse Engineering WCRE'95* (Toronto, 14-16 juillet 1995), IEEE Computer Society Press, Los Alamitos, 126-135.

Elwahidi, A. & Merlo, E. (1995). Generating User Interfaces from Specifications Produced by a Reverse Engineering Process, in *Proceedings of the 2nd Working Conference on Reverse Engineering WCRE'95* (Toronto, 14-16 juillet 1995), IEEE Computer Society Press, Los Alamitos, 292–299.

Furnas, G.W. (1986). Generalized Fisheye Views, in Proc. of ACM Conference on Human Factors in Computing Systems CHI'86 (Boston, 13-17 avril 1986), ACM Press, New York, 16–23.

Hainaut, J-L., Englebert, V., Henrard, J., Hick, J-M., & Roland, D. (1995). Requirements for Information System Reverse Engineering Support, in *Proceedings of the 2nd Working Conference on Reverse Engineering WCRE'95* (Toronto, 14-16 juillet 1995), IEEE Computer Society Press, Los Alamitos, 136-145.

Kong, L., Stroulia, E., & Matichuk, B. (1999). Legacy Interface Migration: A Task-Centered Approach, in *Proceedings of the 8th International Conference on Human-Computer Interaction HCI'International'99* (Munich, 22-26 août 1999), H.-J. Bullinger & J. Ziegler (éds.), Lawrence Erlbaum Associates, Vol. 1, 1167–1171, accessible à http://www.cs.ualberta.ca/~stroulia/Papers/hci99.ps

Moore, M.M. & Rugaber, S. (1997). Using Knowledge Representation to Understand Interactive Systems, in *Proceedings of the IEEE 5th International Workshop on Program Comprehension IWPC'97* (Dearborn, 28-30 mai 1997), IEEE Computer Society Press, Los Alamitos, accessible à http://www.cc.gatech.edu/fac/Melody.Moore/papers/WPC97

Moore, M.M., Rugaber, S., & Seaver, P., (1994). Knowledge Based User Interface Migration, in *Proceedings of the 1994 International Conference on Software Maintenance*, (Victoria, septembre 1994), accessible à http://www.cc.gatech.edu/reverse/repository/uif_migration.ps.

Pemberton, S. (26 janvier 2000). XHTML™ 1.0: The Extensible HyperText Markup Language: A Reformulation of HTML 4 in XML 1.0, W3C Recommendation, World Wide Web Consortium, accessible à http://www.w3.org/TR/xhtml1/.

Puerta, A.R. & Eisenstein, J. (2001). Representational Basis for User Interface Transformations, in *Proc. of ACM CHI'2001 Workshop on Transforming the UI for Anyone. Anywhere* (Seattle, 1-2 avril 2001).

Puerta, A.R. & Eisenstein, J. (2002). XIML: A common Representation for InteractionData, in Proc. International Conference on Intelligent User Interfaces, Januar 13-16, San Francisco, California, USA, ACM Press, accessible à http://www.iuiconf.org/02pdf/2002-002-0043.pdf

Sahuguet, A. & Azavant. F. (1999). Looking at the Web through XML glasses, in *Proc. of 4th International Conference on Cooperative Information Systems CoopIs'99* (Edimbourg, 2-4 septembre, 1999), SIGMOD Digital Symposium Collection Editor, accessible à http://db.cis.upenn.edu/W4F/publi.html

Stroulia, E., El-Ramly, M., Sorenson, P., & Penner, R. (2000). Legacy Systems Migration in CelLEST. Short Research demonstration, in *Proceedings of the 22nd International Conference on Software Engineering* (Limerick, 4-11 juin 2000), accessible à http://www.cs.ualberta.ca/~stroulia/Papers/rd-final.pdf

Stroulia, E. & Kapoor, R.V. (2002). Reverse Engineering Interaction Plans for Legacy Interface Migration, Chapter 26, in C. Kolski & J. Vanderdonckt (eds.), *Proceedings of 4th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2002* (Valenciennes, 15-17 May 2002), Kluwer Academics Pub., Dordrecht, 295–310.

Sussman, J., Banavar, B., & Bergman, L. (2001). Generalization: A Key Concept in the Creation of Platform Independent User Interfaces, in *Proceedings of Workshop on Application Models and Programming Tools for Ubiquitous Computing UbiTools 2001* (Atlanta, septembre 2001).

Szekely, P. (1996). Retrospective and Challenges for Model-Based Interface Development, in *Proc. of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96* (Namur, 5-7 juin 1996), J. Vanderdonckt (ed.), Presses Universitaires de Namur, Namur, 1996, pp. xxi-xliv.

Thevenin, D. & Coutaz, J. (1999). Plasticity of User Interfaces: Framework and Research Agenda, in *Proc. of 7th IFIP Conference on Human-Computer Interaction Interact'99* (Edimbourg, 30 août–3 septembre 1999), A. Sasse & C. Johnson (éds.), IFIP IOS Press Publ., 110–117.

Vanderbrandt, M.G.J., Klint, P., Verhoef, C., Reverse Engineering and System Renovation – an Annoted Bibliography. Centre for Mathematics and Computer Science, University of Amsterdam, accessible à http://adam.wins.uva.nl/~x/reeng/REanno.html

Vanderdonckt, J. & Bodart, F., (1993). Encapsulating Knowledge for Intelligent Interaction Objects Selection, in *Proc. of ACM Conf. on Human Aspects in Computing Systems InterCHI'93* (Amsterdam, 24-29 avril 1993), ACM Press, New York, 1993, pp. 424-429.

Vanderdonckt, J., Bouillon, L., & Souchon, N. (2001). *Flexible Reverse Engineering of Web Pages with VAQUITA*, in *Proceedings of IEEE 8th Working Conference on Reverse Engineering WCRE'2001* (Stuttgart, 2-5 octobre 2001, IEEE Press, Los Alamitos, 241–248.