Retargeting Web Pages to other Computing Platforms with VAQUITA

Laurent Bouillon and Jean Vanderdonckt

Université catholique de Louvain, Institut d'Administration et de Gestion
Place des Doyens, 1 – B-1348 Louvain-la-Neuve, Belgium
E-mail: {bouillon, vanderdonckt}@isys.ucl.ac.be

Abstract

Mobile platforms are becoming an increasingly important alternative for accessing web pages. Many Web pages are not suited to these platforms and need to be adapted, or rewritten from scratch. Adaptation can be made in two ways: either by a dynamic conversion or by a static reengineering. VAQUITA belongs to the second category by applying a model-based approach to reverse engineer web pages at a certain level of abstraction to transfer them to other computing platforms afterwards. Instead of reverse engineering a presentation model and translating it into another model specified for a particular platform, this paper proposes a reverse engineering tailored for any target platform, even one not yet defined. The essence of this reverse engineering approach consists in composing several functions of abstraction, reflection, translation, and reification into two steps: retargeting and regenerating a web page to another platform.

Keywords: Forward engineering, heuristics, knowledge bases, interface migration, mapping rules, model-based approach, presentation model, reverse engineering, static analysis, web page, WML, XML, XHTML.

1. Introduction

The market for mobile platforms (e.g., PalmPilot, Pocket PC, laptop, mobile phone, SmartPhone, ScreenPhone) is growing two times faster than the market for traditional desktop computers [3]. As the expansion rate of mobile platforms is becoming greater and their capabilities have considerably improved, there is now a need to adapt rapidly the huge amount of web pages in order to make them accessible to a variety of contexts of use (e.g., different devices in different environments). In the reminder of this paper, the context of use will focus on the computing platform.

There are two main approaches to achieve the conversion of a user interface (UI) from one platform to another: either transcoding [4] dynamically performed at runtime or reengineering with a model-based approach along with a static, dynamic or behavioural analysis. Representative examples of this approach are [8,11,12,13,14,18,22]. The knowledge contained in the models [19] is captured in a reverse engineering process that can later be used in the forward engineering phase to produce a new UI.

This paper proposes a new method to convert HTML pages into other formats thanks to a model-based approach: *retargeting*. Retargeting can be used to limit the number of different presentation models produced by traditional reverse engineering, and help the designer in selecting more appropriate models for a specific platform.

An interface created for a particular platform can be viewed as a collection of interrelated widgets called concrete interaction objects. They are said to be concrete because they are dependent on their running environment.

As the relation between an abstract interaction object (AIO)-platform independent- and a concrete interaction object (CIO) is of the type many-to-many [1,16,21], the number of different abstract presentations produced by a "classical" reverse engineering can be relatively large.

In fact, a CIO can be translated by every AIO belonging to the same class. For example, a group of concrete radio buttons can be translated by a group of checkboxes where only one choice is possible, a drop down list box, a textbox, a group of buttons or by a group of abstract radio buttons. Two major constraints reduce this number of possible presentations: the domain and the platform.

The domain contains the data that has to be displayed on AIOs and therefore it restricts the number of possibilities, by the fact that an AIO cannot represent every type of information. The "nature of a domain object is the determining factor in how to present and make the object available to the user" [16]. The second constraint is the platform, on which a predefined set of AIO is available. Further (weaker) constraints can later be used in order to select the most appropriate AIO, such as the space consumption, the ease of use, the "look and feel" of the application In our approach, as we start with the presentation model, we do not have information about the domain model, but it is possible to reverse engineer an interface for a pre-selected computing platform and so limit the number of possibilities.

The remainder of this paper is structured as follows: section 2 provides related works used to access an HTML page from other devices and similar reverse engineering approaches, section 3 explains the reference framework in which this paper is situated, section 4 is dedicated to the reverse engineering process as supported in VAQUITA, section 5 contains a case study on which the proposed method is applied, section 6 concludes this paper and proposes future work.

2. Related Work

Two approaches exist for converting HTML pages to be accessed by other devices: direct transcoding (dynamic) and model-based approaches (MBA). Among the dynamic methods, the *Digestor* [2] software system provides device-independent access to web pages by focusing on the reduced I/O capabilities of mobile platforms. This problem is tackled by elision techniques. For example, in a large text document, the headers are elided and transformed into links that give access to the content of the section. An algorithm applies a series of 15 transformations with space consumption as performance measure after each transformation step. If the space reduction is unsatisfactory, another transformation technique is tried.

The approach proposed in [10] similarly applies a set of direct transformations to the HTML objects. Like Digestor, the transformations are made through an HTTP proxy. Digestor applies elision techniques whereas elements [10] are transformed into other types or cut off.

IBM *Websphere* [4] uses several techniques to render HTML on mobile phones: HTML documents simplification to tailor them for reduced function devices, HTML to WML transcoding (syntactically), source identification (what type of mobile phone), extensible style sheet selection and application followed by a deck fragmentation and image conversion. The complete transcoding is done at the server level. As all these techniques are at run-time, no (or few) choices are possible regarding the transformations. In our approach, we emphasize the freedom left to the user to control the transformation of the objects.

For static methods, we are not aware of MBA in the conversion of Web pages to other computing platforms. However, several MBA exist for generating Web pages.

Ware [7] is a reverse engineering tool for Web applications. Ware combines static and dynamic analyses to capture the information necessary to construct UML diagrams (class, use case, sequence). Ware is a support tool for web site maintenance and evolution.

Reweb's reengineering process [17] restructures Web applications in order to avoid their inevitable degradation. It uses a set of transformation rules aiming at the improvement of maintainability, usability and portability. It also restructures the design thanks to a web application model and by incorporating frame-based navigation.

ArtStudio [5,20] generates user interfaces in several languages and for different platforms. ArtStudio automatically generates UIs in four steps by using a platform model, an interactor model (list of the available widgets on a platform), an environment model, an evolution model, a task model and concepts (domain) model. This approach also enables context-sensitive UIs to be built by taking environment and platform information into account. A more exhaustive list of reverse engineering and transcoding methods can be found at http://www.isys.ucl.ac.be/bchi/research/soare.htm.

3. Reference Framework

This section is divided into four subsections: the reference framework for context sensitive UIs, the extensions of the framework for the purpose of this paper, the mathematical definition of retargeting and the presentation model, which is the result of the reverse engineering.

3.1 A framework for context sensitive UIs

Context sensitive UIs are interfaces which can adapt their composition, appearance, structure, and behavior according to the context in which they are used. A context of use [14] is defined as the triple "user, platform, environment". In our case, only one element is subject to variations: the platform. In this framework [20] developed by D.Thevenin (and subsequently expanded in [6]), an interface can be decomposed into four levels (fig. 1):

- **Concepts and Task Model:** contains the description of the various tasks and the domain-oriented concepts related to these tasks. Let TC denote the set of possible task and concepts.
- **Abstract Interface:** defines working spaces [5] by grouping subtasks according to various criteria (e.g. cognitive load, semantic relationships, shared concepts), a navigation scheme between the working spaces and selects Abstract Interaction objects for each concept. Let AI denote the set of abstract UIs.
- **Concrete Interface:** is a realization of the abstract interface for a given target. At this level, interactors, layout of objects and navigation within the interface are defined. There is no link with the semantic core for this interface. Let CI denote the set of concrete UIs.
- **Final interface:** is the operational interface. At this step, the UI code is generated. Let FI denote the set of possible final user interfaces.

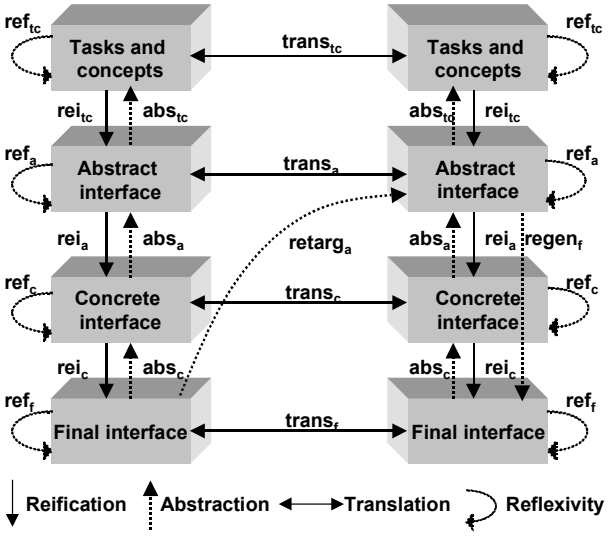

Figure 1. The reference framework for user interfaces.

The downward arrows represent reification steps, from the more abstract to the operational interface. Horizontal arrows correspond to the translation of a UI from one type of platform to another, or more generally, from one context to another. Loops graphically depict reflexive function applied at a same level of abstraction on a same UI.

3.2 Extensions of the Framework

Two new types of operation were introduced in this framework: *abstraction* and *retargeting*. Abstractions are represented as upward arrows in fig. 1 and stand for the reverse engineering steps. In this framework, the approaches proposed by [2], [4] and [17] are situated on the bottom translation arrows ($trans_f$), as the transformation of objects is applied at the concrete level. Artstudio[20] follows the downwards path from tasks and concepts to the final interface. Ware [7] and Reweb [17] are situated at the abstraction from final interface to the concrete interface (abs_c). In Reweb, the backward reification step (rei_c) are also done after the transformations done at a higher level of abstraction.

The usual way for reverse engineering to proceed is to go up to the abstract interface point (abs_c and abs_a), and translate it into another abstract interface which is specialized for a particular platform ($trans_a$). This method may generate several presentation models, at each step of the process (fig. 2).

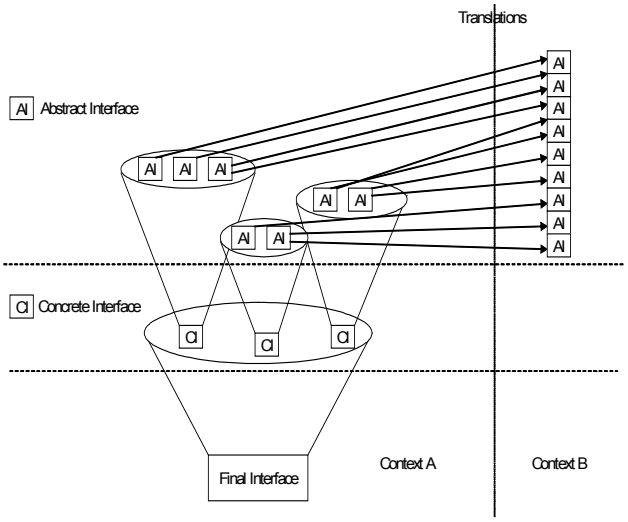

Figure 2. Proliferation of presentation models.

The underlying idea in this article is that, if the target platform is known before the reverse engineering, the resulting presentation model will be of better quality, and this process will help the designer by reducing the number of possible results.

For the translation step, there are two possibilities: either the target platform is more constraining than the source platform, and in this case the translation will reduce the number of possibilities (e.g., the translation from a PC UI into a mobile phone UI), or the target platform is less constraining, and the translation will increase the number of possible presentation models (e.g. the translation from a Palm UI into a Macintosh UI). The retargeting process is represented by the large dotted line in fig. 1. As shown, the reverse engineering is directly applied to the target platform.

3.3 Mathematical Definitions

The purpose of this section is to introduce a mathematical definition of the UI transformations performed on each level of the extended reference framework. Each UI transformation at any level of abstraction in fig. 1 can be denoted as a mathematical function. A *mapping* M:A→B is represented as a set

$$M = \{(x,y) \dagger x \in A \text{ and } y \in B\}$$

where A is called the *domain of* M, written *domain(*M*)* and B is called the *range of* M, written *range(*M*)*. A *m-to-1* mapping M associates one or more than one element in *domain(*M*)* with each element in *range(*M*)*. An *onto* mapping M associates elements of A with all elements of B. A *function f* is then defined an a *m-to-1* and *onto* mapping. If *f* is a function, then the range of *f* is also called the co-domain of *f*, written *co-domain(f)*.

Reification. Reification functions are functions mapping a UI representation from one non-terminal level of abstraction to a lower level of abstraction. The functions rei_{tc}, rei_a , rei_c respectively produce an abstract interface from a couple (task, concepts), a concrete interface from an abstract interface and a final interface from a concrete:

Abstraction. Abstraction functions are functions mapping a UI representation from one non-initial level of abstraction to a higher level of abstraction. The functions abs_c, abs_a, abs_{tc} respectively produce a concrete interface from a final interface, an abstract interface from a concrete interface, and a couple (task, concepts) from an abstract interface.

Translation. Translation functions are functions mapping a UI representation at a same level of abstraction, but from one context of use to another. The functions $trans_{tc}$, $trans_a$, $trans_c$, $trans_f$ respectively produce a couple (task, concepts), an abstract interface, a concrete interface and a final interface at the same level of abstraction for another context of use. For example, the translation function at the tasks and concepts level is defined by:

$trans_{tc}$: TC X C X C→TC: (t_1,d_1) x c_1 x c_2 ↶ (t_2,d_2)
where c_1, c_2 respectively denote the initial and final context of use and d_1, d_2 represent concepts.

Reflexivity. Reflexive functions are functions mapping an existing UI representation at a given level of abstraction to another UI representation at the same level of abstrac

tion for the same context of use. For example, the reflexive function defined at the abstract UI level is defined by:

ref_a: AI X C →AI: ai_1 x c ↷ ai_2
where ai_1, ai_2 respectively denote the initial and resulting abstract interfaces and c represents a context.

Composition of functions. As it turns out to be important to perform functions one after another, to combine functions in sequence, the composition of functions needs to be defined. If $f:A\rightarrow B$ and $g: B\rightarrow C$ are two functions, then the *composition of f and g*, written as f o g, is defined as:

$$f \text{ o } g = \{(x,y) \dagger \forall x \in A, \exists y \in C: y=g(f(x)) \}$$

that is f o g = {(x,y) † ∀ x∈A, ∃ y∈C, z∈B such as (x,z)∈B and (y,z)∈C}. In this definition, f and g can be composed because *domain(g)=co-domain(f)*. The reverse composition g o f is not necessarily valid since *domain(f)* ≠ *co-domain(g)*.

Inverse functions. Another important concept in this study of functions is inverse functions since we want to perform both forward and reverse engineering of UIs. In addition, it is important to identify when a particular UI produced by forward engineering (i.e., by applying reification, translation, reflection or any combination of them) can be completely recovered by reverse engineering (i.e., by applying abstraction, translation, reflexivity or any combination of them). The same question arises with respect to a UI produced by reverse engineering to be reproduced by forward engineering. These conditions can be expressed in term of inverse functions.

Let f and g be two functions. If f(g(x)) = x and g(f(x)) = x, then g is the *inverse of f* and f is the *inverse of g*. The inverse function of a function f is denoted f^{-1}. In this case, f is said to be *inversible*. To ensure a true bidirectional engineering of UI at any level of abstraction, all involved functions should be inversible, corresponding conditions must hold. For the abstraction and reification functions, at each abstraction level, we have:

∀ (t,d)∈TC, ∃ a∈AI † $rei_a(t,d) = a$ and $(t,d) = abs_{tc}(a)$
∀ a∈AI, ∃ c∈CI † $rei_a(a) = c$ and $a = abs_a(c)$
∀ c∈CI, ∃ f∈FI † $rei_c(c) = f$ and $c = abs_c(f)$

For example, any concrete interface can be reified into a final interface, which can be in turn abstracted into its original form at the concrete interface level by the corresponding inverse function: abs_c

For the translation functions (context change), at each abstraction level, we have:

∀ (t_1,d_1)∈TC, ∀ c_1,c_2∈C, ∃ (t_2,d_2)∈TC † $trans_{tc}(t_1,d_1), c_1, c_2) = (t_2,d_2)$ and $trans_{tc}(t_2,d_2), c_2, c_1) = (t_1,d_1)$
∀ a_1∈AI, ∀ c_1,c_2∈C, ∃ a_2∈AI † $trans_a(a_1, c_1, c_2) = a_2$ and $trans_a(a_2, c_2, c_1) = a_1$
∀ d_1∈CI, ∀ c_1,c_2∈C, ∃ d_2∈CI † $trans_c(d_1, c_1, c_2) = d_2$ and $trans_c(d_2, c_2, c_1) = d_1$
∀ f_1∈FI, ∀ c_1,c_2∈C, ∃ f_2∈FI † $trans_f(f_1, c_1, c_2) = f_2$ and $trans_f(f_2, c_2, c_1) = f_1$
where C denotes the set of possible contexts.

For the reflection functions, at each abstraction level, these properties are written as follows:

∀ (t_1,d_1)∈TC, ∀ c∈C, ∃ (t_2,d_2)∈TC † $ref_{tc}(t_1,d_1), c) = (t_2,d_2)$ and $ref_{tc}(t_2,d_2), c) = (t_1,d_1)$
∀a_1∈AI,∀ c∈C,∃ a_2∈AI † $ref_a(a_1,c)=a_2$ and $ref_a(a_2, c)=a_1$
∀c_1 ∈ CI,∀ c∈C,∃ c_2∈CI † $ref_c(c_1,c)=c_2$ and $ref_c(c_2, c)= c_1$
∀ f_1∈FI, ∀ c∈C, ∃ f_2∈FI † $ref_f(f_1,c) = f_2$ and $ref_f(f_2, c) = f_1$
where C denotes the set of possible contexts.

Restriction. For a given function to be applied in a specific computing platform, there is a need to define a condition to be satisfied when applying this function. For example, a constraint may be imposed when a translation between two computing platforms occurs, such as: "the target computing platform does not allow hierarchy of presentation elements deeper than a certain threshold". The WML language instantiates this constraint to 9 (not more than 9 presentation levels in decks and cards), while certain versions of cHTML instantiates this contraint to 4 (not more than 4 levels of cHTML tags). The restriction of function is therefore required.

A *restriction* (or *selection*) of a function f:A→B, denoted as $\sigma_{cond}(f)$, is a function such that

$\sigma_{cond}(f)$ = {(x,y) † ∀ x ∈ A, ∃ y ∈ B: y=f(x) and cond(x,y) = true} where cond is any first-order predicate.

Retargeting. Retargeting can now be defined as a shortcut by a composition of three functions restricted by the conditions imposed by the specific target computing platform:

$$retarg_a = \sigma_{cond}(trans_a \text{ o } abs_a \text{ o } abs_c) \text{ and } cond = \bigwedge_{i=1}^{n} c_i$$

where *cond* is a set of first-order predicates expressing constraints (c_i) applicable for a particular target (e.g., NumberOfPresLevels < 10 ∧ NumberMenuItems < 7).

Regenerating. To complete the process to obtain another UI for any other computing platform, regenerating can now be defined similarly by the composition $regen_f = rei_c$ o rei_a so as to represent the complete process by $regen_f$ o $retarg_a$.

Idempotence. Some functions have the special property that applying them more than once to the same co-domain produces no further change after the first application. For instance, $f(f(x)) = f(x)$. To ensure a true bidirectional engineering of UI at any level of abstraction, the composition of all the functions involved and their corresponding function should be idempotent, that is the following conditions must hold:

$f = (rei_c \circ abs_c) \Rightarrow f$ is idempotent since $\forall\ fi \in FI$, $\forall$ i $\in$ {1,…,}: $fi = f(fi) = f(f(\text{fi})) = \ldots = f_{\ldots \text{i times}}(f(\text{fi})) = f \circ f \circ_{\text{i times}} f(\text{fi})$. Similarly, $g = (rei_c \circ abs_a)$ and $h = (rei_a \circ abs_{tm})$ are idempotent.

The inversibility and idempotence of composition as introduced above guarantee a true bidirectional engineering of UI. We can observe in this case that $domain(rei_c) = co\text{-}domain(rei_c^{-1}) = co\text{-}domain(abs_c)$ and $co\text{-}domain(rei_c) = domain(rei_c^{-1}) = domain(abs_c)$. If the properties of inversibility and idempotence no longer hold, then $co\text{-}domain(\text{abs}_c^{-1}) \subset co\text{-}domain(\text{rei}_c) \Rightarrow abs_c^{-1}$ and rei_c are not two inverse functions of each other, but abs_c^{-1} is a restriction of rei_c. The same reasoning can be established analogously for the other levels of abstraction and for the other functions.

It is conceivable that all the above functions can be performed *manually* if and only if it requires a human agent, *automatically* if and only if it requires a computer agent, and *in a mixed-initiative* if and only if it requires both a human agent and a computer agent to collaborate. Moreover, these functions can be performed in a hybrid manner when a third party can intervene in the process. The goal of VAQUITA as presented in this paper is to provide a mixed-initiative support for UI retargeting that allows the same UI to be retargeted for another context of use (here, limited to another computing platform).

However, it can be easily imagined that some of these functions need to be performed manually when human expertise is required (e.g., for adaptation, tailoring, semantic reconfiguration) or automatically when the system can embody enough knowledge to perform them. In particular, reflection functions are a good candidate for fully automatic/mixed-initiative functions: at design time, the system suggests that alternative designs be explored.

Retargeting is done by translating the concrete interaction objects in the source UI into the chosen (by the user) abstract interaction object. There are four main types of targeting operations:

- Translate an interaction object of the source page into an object type specific to the target platform. For example, it would be possible to translate an image-map into a group of text-links or buttons when the target platform has no mouse device.
- Modify the value of some attributes of the original interaction objects. For example, for a monochrome platform, every colour attribute in each object has to be modified into a black or white colour.
- Delete an element that cannot be displayed on the target computing platform. For example, for a platform without sound capabilities, the background sounds of an HTML page should be discarded.
- Transfer (Move) of interaction objects: platforms with smaller display capabilities than the source imply that the layout and page composition have to be modified. Some interaction objects can be moved from one logical window to another, or new logical windows can be created and filled with transferred elements in order to cope with the available screen surface.

For the moment, retargeting is applied only to the presentation model, but the process could be extended to the other models, especially the dialogue, so that the two mandatory models to generate an interface would be completed.

3.4 The Presentation Model

A presentation model [9] [16] [22] contains every element provided by a UI to its users, such as buttons, checkboxes or background music. The model is composed of several elements ordered hierarchically:

- *Concrete Interaction Objects (CIOs)* are the real objects that belong to the Final Interface.
- *Abstract Interaction Objects(AIOs)* are the abstract counterparts of the CIOs (e.g. without a hardware or a software reference). Each AIO can represent 0, 1 or many CIOs[1],[22].
- *Simple* AIOs are objects that cannot be divided. An AIO is said to be *Composite* when it is composed of simple or other composite AIOs.
- *Logical Windows* are either logical containers structuring the presentation model, or a physical container object such as a panel, a window or a dialog box.

All these concepts are represented in the presentation model as embedded presentation elements. The position in the hierarchy where the presentation element takes place determines its type.

4. The reverse engineering process in VAQUITA

VAQUITA (reVerse engineering of Application by Questions, Information Selection and Transformation Alternatives – http://www.isys.ucl.ac.be/bchi/research/vaquita.htm) is an application that enables the reverse engineering of the presentation model for HTML pages [22]. The reverse engineering is done by applying mappings rules in a static analysis of one HTML page at a time.

The complete process in which VAQUITA takes part is represented on fig.3. The page is first saved locally and its XML compliance is checked. If the page does not respect the XML grammar, the HTML page is syntactically corrected thanks to the TidyCom library [23].

The use of this library offers several advantages: first, it ensures us to always have the same code format; secondly it cuts useless tags off (such as empty paragraphs:), so that the process time of VAQUITA is reduced; and finally it transforms the HTML into the XML format, allowing us to use the msxml3 library. The cor

rected page is then parsed with the help of the msxml3 library [24]. This library offers multiple functions for markup languages, such as tag retrieval, creation, modification, etc. The DOM structure is then extracted and the targeted mapping rules are applied.

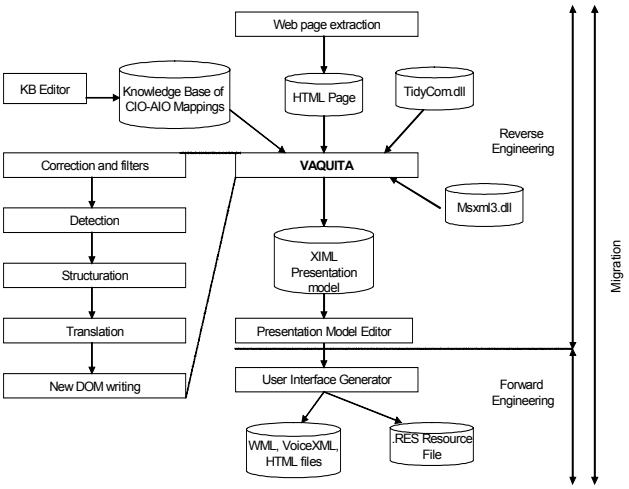

Figure 3. The migration process using VAQUITA.

These rules are loaded at the beginning of the reverse engineering process. The CIO to AIO mapping rules are stored in a knowledge base, with one separate file for each platform. These mappings are the more probable ones, but a specific editor allows one to modify or add new rules easily, so that designers can customize mapping rules for a specific Web site. An example of the rules, stored in the XML format, is shown below:

```
<Object name="RadioButton">
  <Transf name="CheckBox">
      <attribute name="DefaultValue">DefaultValue
      </attribute>… </transf>
   <Transf name="Listbox">
      <attribute name="NumberOfButtons>Size
      </attribute>
      <attribute name="Caption_Label">ListItem
      </attribute>… </transf>
```

This simplified example shows the alternatives transformations for a radio button, here a checkbox and a listbox: for each attribute of the radio button, a corresponding attribute from the new object has to be specified. Transformations of attributes values are not yet allowed, but this is a future work. As one can see on this example, captions of radio buttons are used in the listbox transformation as list items. But these labels are not linked with their corresponding objects in the HTML code. The deduction of these relations can be done in two ways: either automatically by using heuristics. In this case, unsolved cases are let empty, and will be resolved manually after the reverse engineering. The second way is the assisted method, in which the system proposes a set of choices for each label and so allows finishing the reverse engineering with a more complete presentation model. The presentation model is written in the eXtensible user-Interface Markup Language [25] a language developed by RedWhale Software Corp. (www.redwhale.com), derived from XML and able to store the models developed in MIMIC[15].

This model can still be modified thanks to an XIML editor, in order to refine the results obtained with VAQUITA. In [22], a previous version of VAQUITA was presented, but this older version only allowed the flexible reverse engineering of Web Pages. The added features in this new version are:

- The possibility to target the reverse engineering to a computing platform, so translations in the reference framework (section 3.1) are added to the capabilities of the tool.
- A Knowledge Base contains a series of individual conditions that can be grouped to restrict the retargeting according to the target computing platform.
- A dedicated editor enabling the CIO to AIO mapping rules to be quickly modified
- The incorporation of the Tidy tool [26] in the application, under the form of a library, permitting the user to directly reverse engineer the web page after having saved it locally.

5. Case Study

This case study shows the targeting of an HTML page for a mobile phone. In the first part of this section, a short description of the problem is shown. The second subsection gives some examples of choices in a targeting from HTML to mobile phones. The third part is dedicated to the targeting of the presentation model of the Hotmail Home Page (http://www.hotmail.com) and in the last one, a brief comparison with traditional reverse engineering is shown.

5.1 Problem description

Transforming an HTML page for small sized platforms is a challenging process that is composed of two types of problems: first, the set of available interaction objects is different, thus some transformation of the UI are needed. Secondly, the small size of the screen and the limited bandwith forces the designer to reduce the content of the original page. Because of this second category, the reengineering of web pages '*in the small*' (i.e. to platforms with less capabilities) will never be fully automatic. Indeed, the fact that some parts are more important than others can not be inferred by VAQUITA (or another tool). This is thus a mixed initiative (human-computer) process and the size reduction step is done in our approach thanks to a dedicated tool (Section 4) on the retargeted presentation model.

5.2 Tool Usage - Alternatives in the reverse engineering

The retargeting of HTML to WML offers multiple transformations that have to be chosen by the designer. For each target platform, a list of possible transformations is available to the user and the most probable one is already pre-selected. An example for the translation of an image in the context of the retargeting for a mobile phone is given in table 1.

Images	Transform into a label composed of the alternative text
	Transform into a text-link that redirects to the image
	Convert it to the .WBMP format
	Suppress all images
	Suppress images bigger than x pixels
	Suppress images with a size lower than 50 bytes

Table 1. Alternatives in the transformation of images.

There are six available transformations for images in WML. The designer can choose to keep the image - in the .WBMP format - with a direct access or not (in this latter case, the end-user will choose if it is worth losing connection time to see the image). The designer can also put constraints on the transformation: he can choose a threshold of size (i.e. the image would become unclear on a small screen) or to skip small-sized images (usually, these images are only used to have a stable layout in HTML).Table 2 shows another example of transformation alternatives for text-links.

Text Links	Keep only intra-site links
	Transform intra-page links into extra-page links
	Suppress download-links
	Suppress redundant links
	Suppress style attributes
	Suppress colour attributes

Table 2. Different alternatives for text links.

Text-links are divided into 3 categories: internal or external to the web site, internal or external to the page and navigation or download links. The designer can choose to suppress links external to the web site, in order to keep only the really important links for the navigation of the Web site. (S)he can also choose to transform intra-page links into extra-page links: intra-page links are generally used to jump to precise parts of long web pages that are not suitable for small-display platforms (huge scrolling manipulations). This problem can be reduced by breaking the page into smaller ones, and this subdivision does not disturb the user's perception of the site structure. When this option is chosen, the rest of the page is automatically transformed into a set of smaller pages (see fig. 4).

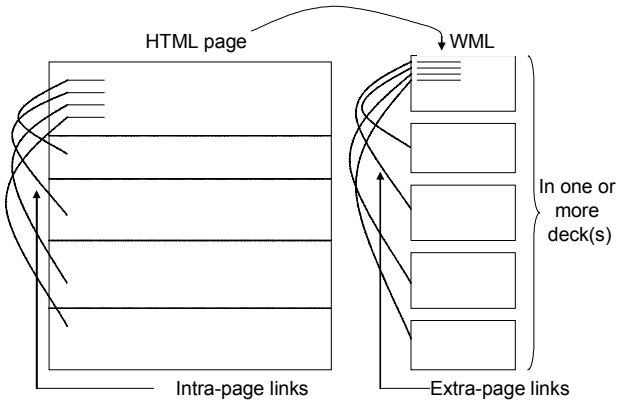

Figure 4. The transformation of intra-page links into extra-page links.

Since mobile phones do not have a secondary memory, download links are useless on this platform and can be removed. Sometimes, we can find the same target for different links on the same page. These redundant links can be automatically removed to save screen space. And finally, the designer can choose to suppress the colour and the style (e.g., bold, italics) attributes for each link. These two last options are available for each textual object (e.g., links, labels, marquees), but can also be selected at the model level in order to avoid repetitive manipulations in VAQUITA.

These different transformation alternatives are available through the options pages in VAQUITA. On figure 5, the different possibilities for an HTML table are shown.

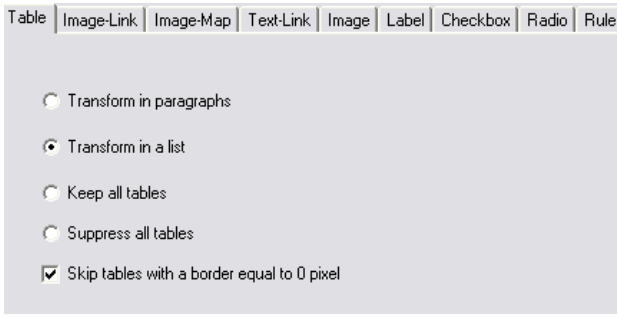

Figure 5. The transformation options in VAQUITA.

The designer can choose to convert automatically every table into several paragraphs, or to transform it into a list. The designer can add one constraint for the analysis of tables: to analyse only tables with a visible border (invisible tables are only used in HTML to organize the page). Further refinements can be made on the detection and transformation of attributes on the general detection options page. For the first example of images, if the designer had chosen to transform them into the .WBMP format, attributes such as image height or width are automatically unchecked on the option page (see fig.6). If (s)he wants to keep this information in the presentation model, he can still check the chosen attributes again.

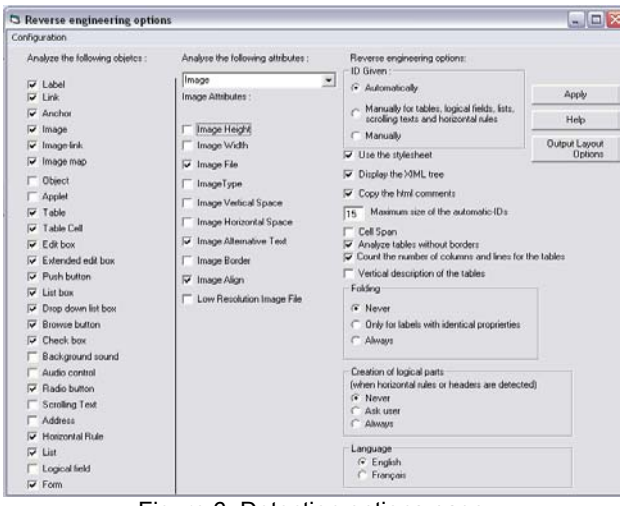

Figure 6. Detection options page.

The user is given the freedom to to re-modify the detection of the elements and their attributes after the selection of the translation rules.

5.3 Result description

The Hotmail page (fig. 7) is composed of 12 different CIO types and contains 71 objects. This page contains a variety of interaction objects, is relatively small and supports heavy traffic.

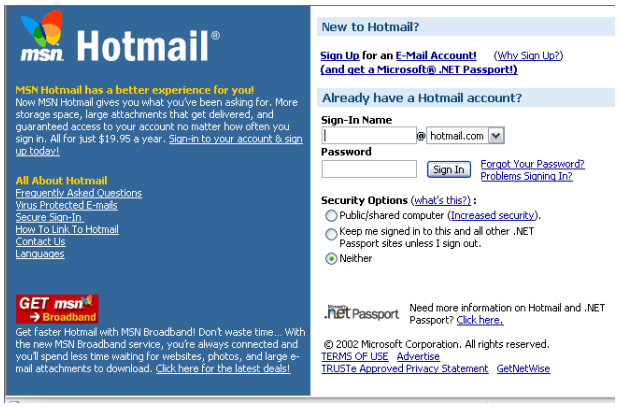

Figure 7. The Hotmail home page.

The transformations done in this targeting are mostly the made-up conversion of images into labels of their alternative texts, suppression of unneeded elements and the conversion of color attributes into monochrome colors. However, some transformations are more elaborate. For example, there is a group of radio buttons linked with their labels. This is represented in XIML thanks to relation "label_describes". But radio buttons are not supported in WML, and so are converted into a Boolean selection list, which is composed of the corresponding labels. There is also a link included in these labels. Because links are not allowed in a selection list, the link is moved and placed after the selection list.

The presentation model of Hotmail targeted for mobile platforms is much smaller than the presentation model without targeting, because less information is needed for this type of platform. The division into logical windows is done by calculating the memory used by a sequence of elements, until this sum becomes greater than a fixed threshold. The ordering of the logical windows (that will later correspond to cards) is the sequence of elements in the HTML page. This division is, of course not, optimal; it would be better, for example, to start the WML deck with the login and password input boxes than by Hotmail logo and descriptive labels (upper left corner of figure 7). Such modifications can be done at the redesigning phase of the presentation model, thanks to a XIML editor.

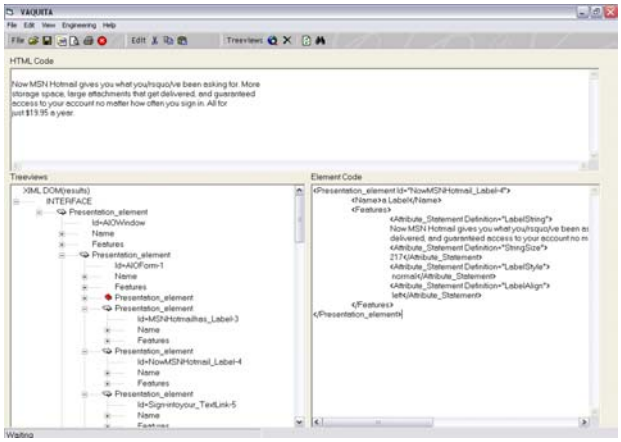

Figure 8. VAQUITA main screen.

In fig. 8, a screenshot of VAQUITA's main screen after targeting is shown. This screen is composed of three different views: on the left, the user can switch two treeviews. The first tree corresponds to the HTML file and the second represents the hierarchical structure of the presentation model. In fig.8, the presentation element 'NowMSNHotmail-Label' is selected in the presentation model. The XIML code associated with this presentation element and its attributes are displayed on the right, and the corresponding HTML code appears on the top of the screen. In fig.9, one can see the result for a mobile phone of the forward engineering phase based on the presentation model.

Figure 9. Hotmail on a mobile phone.

5.4 Comparison with reverse engineering

Differences between a presentation model obtained by traditional web page reverse engineering with VAQUITA without and with targeting are shown in table 2.

	Rev.Eng.	**Targeting**
File Size	59.9 ko	39.1 ko
Number of AIO	71	53
Time	113 seconds	106 seconds
Correctness	-	Better
Domain cardinality	High	Moderate
Cognitive load	High	Moderate

Table 3. Comparison between reverse engineering and retargeting processes.

It should be noticed that reverse engineering followed by a translation will suppress elements. These suppressed elements are the target or the source of layout relationships (or other types of relations), and these ones will become false. Time to correct this point is not comprised in the table. The correctness of the targeted page is of better quality for the same reason (the fact that elements are transformed and/or suppressed before the inference of the relationships provides more information for the deduction process). The cardinality of the domain in a reverse engineering is also much higher as for targeting, by the fact that there are less constraints for transformations (and thus possible solutions) in the reverse engineering approach. For the same reason, the cognitive load in the reverse engineering is also higher than for the targeting process.

6. Conclusion and future work

In this paper, we have presented a method and a tool allowing designers to reverse engineer a presentation model targeted for a particular platform. Targeting transformations are applied in a static analysis and are stored in a knowledge base editable by the user. The advantages of targeting over a reverse engineering are numerous:

- Speed ups and helps the choice of the best fitting presentation model for a chosen platform.
- Generates more correct presentation models (constraints related to the target platform can be tested during the construction of the model; relationships are correct thanks to the knowledge of the remaining elements)
- Enables the user to obtain directly the results wanted, rather than to shrink a model which was too big, thus reducing the process time.

Retargeting is considered more efficient than the composition of functions it covers as it performs the same service while effectively exploiting the UI design knowledge appropriate for a particular computing platform. Retargeting does not introduce any loss of generality since all reverse engineering options supported by abs_c, abs_a, and $trans_a$ respectively are embedded in $retarg_a$. Retargeting is considered faster than the composition of functions it covers since the designer is not forced to apply the functions in a sequence and to check/tailor the results before applying the next function. And, finally, retargeting is considered more robust than the composition since it prepares an abstract UI that is compatible with constraints imposed by another computing platform by construction. The composition of retargeting and regenerating enables designers to obtain virtually any new UI for another computing platform. In this paper, the process has been illustrated between an HTML web page and a WML deck of cards, but the process can be executed to any other computing platform and not just WAP-compatible mobile phones. The process is therefore much more general than any other transcoding approach and any other reverse engineering process conducted at the concrete UI level.

VAQUITA can support UI retargeting to different existing computing platforms at the same time (and not just only one), such as WAP phones with different types of display, Internet ScreenPhones (e.g., Alcatel's WebTouch as described in http://www.webtouch.alcatel.com/), Web appliances (e.g., WebTV), and PocketPC (e.g., through micro-browsers). VAQUITA can even support future computing platforms through the definition of restrictions of functions involved in the retargeting. For instance, if a new computing platform appears that constrain the depth of presentation elements up to 5 levels, this constraint can be logically implemented as a restriction of an existing function to form a new retargeting.

Future studies will be dedicated to the targeting of the dialogue model in order to be able to generate the new interface. For the moment, only WAP-targeting is available, but we will soon add other platforms, such as PDAs. We are also looking at the expansion of the level of generality of the method. We could extend the current process to a higher level of generality, i.e. to all markup languages rather than to the HTML and thus encompassing WML (allowing the reverse engineering of WML). In a further step, this method could be applied to imperative languages, in order to propose a general method for reverse engineering user interfaces.

7. Credits and acknowledgements

The authors would like to thank Pr.Carlon from ILV-UCL for the correction of this paper. This Work is based in whole or in part on the XIML Language and Schema. Trademarks and Copyrights for XIML are owned by RedWhale Software Corp., Palo Alto (CA), USA. XIML is promoted by the XIML Consortium™ (http://www.ximl.org)

8. References

[1] P.J. Barclay and J. Kennedy, "Teallach's Presentation Model", *Proc. of the 5th ACM Int. Working Conf. on Advanced Visual Interfaces AVI'2002* (Palermo, 23-26 May 2000), ACM Press, New York, 2000, pp. 151-154.

[2] T.W. Bickmore and B.N. Schilit, "Digestor: Device-Independent Access to the World-Wide-Web", *Proc. of 6th World-Wide-Web Conf. WWW'6* (Santa Clara, 7-11 April 1997), Elsevier Science Pub., Amsterdam, 1997. Accessible at http://www.scope.gmd.de/info/www6/technical/paper177/paper177.html

[3] J. Berst, J., "Why the world is going mobile", *ZD Net*, 12 December 1998. Accessible at http://www.zdnet.com/anchordesk/story/story_2909.html

[4] K.H. Britton, R. Case, A. Citron, R. Floyd, Y. Li, C. Seekamp, B. Topol, K. Tracey, "Transcoding: Extending E-Business to New Environments", *IBM Systems Journal*, Vol. 40, No. 1, 2001, pp 153-178. Accessible at http://www.research.ibm.com/journal/sj/401/britton.html

[5] G. Calvary, J. Coutaz, D. Thevenin, "A Unifying Reference Framework for the Development of Plastic UIs, *Proc. of 8th IFIP Int. Conf. on Engineering for Human-Computer Interaction EHCI'2001* (Toronto, 11-13 May 2001), R. Little and L. Nigay (eds.), Lecture Notes in Comp. Science, Vol. 2254, Springer-Verlag, Berlin, 2001, pp. 173-192.

[6] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, N. Souchon, L. Bouillon, J. Vanderdonckt, "Plasticity of User Interfaces: A Revised Reference Framework", *Proc. of 1st International Workshop on Task Model and Diagrams for User Interface Design TAMODIA'2002* (Bucharest, 18-19 July 2002), Academy of Economic Studies of Bucharest, INFOREC Printing House, Bucharest, 2002, pp. 127-134.

[7] G.A. Di Lucca, M. Di Penta, G. Antoniol, G. Casazza, "An Approach for Reverse Engineering of Web-Based Applications", *Proc. of 8th IEEE Working Conference on Reverse Engineering WCRE'2001* (Stuttgart, 5-7 October 2001), IEEE Computer Society Press, Los Alamitos, 2001, pp. 231-240.

[8] M. El-Ramly, P. Iglinski, E. Stroulia, P. Sorenson, B. Matichuk, "Modeling the System-User Dialog Using Interaction Traces", *Proc. of 8th IEEE Working Conference on Reverse Engineering WCRE'2001* (Stuttgart, 5-7 October 2001), IEEE Computer Society Press, Los Alamitos, 2001, pp. 208-217.

[9] T. Griffiths, P.J. Barclay, N.W. Paton, J. McKirdy, J. Kennedy, P.D. Gray, R. Cooper, C.A. Goble, P. Pinheiro da Silva, "Teallach: A Model-Based User Interface Development Environment for Object Databases", *Interacting with Computers*, Vol. 14, No. 1, December 2001, pp 31-68.

[10] E. Kaasinen, J. Kolari, T. Laakko, "Mobile-Transparent Access to Web Services", *Proc. of 8th IFIP TC.13 Conf. on Human-Computer Interaction Interact'2001* (Tokyo, 9-13 July 2001), Elsevier Science Pub., Amsterdam, 2001.

[11] E. Merlo, P.-Y. Gagné, and A. Thiboutôt, "Inference of Graphical AUIDL Specifications for the Reverse Engineering of User Interfaces", *Proc. of IEEE Int. Conf. on Software Maintenance* (19-23 September 1994), IEEE Computer Society Press, Los Alamitos, 1994, pp. 80-88.

[12] M.M. Moore, "Representation Issues for Reengineering Interactive Systems", *ACM Computing Surveys*, Vol. 28, No. 4, December 1996. Article # 199. Accessible at http://www.acm.org/pubs/articles/journals/surveys/1996-28-4es/a199-moore/a199-moore.html

[13] M.M. Moore and S. Rugaber, "Issues in User Interface Migration", *Proc. of 3rd Int. Software Engineering Research Forum SERF'93* (Orlando, 10 November 1993). Accessible at http://www.cc.gatech.edu/reverse/repository/ serf.ps.

[14] M.M. Moore and S. Rugaber, "Using Knowledge Representation to Understand Interactive Systems," *Proc. of the 5th Int. Workshop on Program Comprehension IWPC'97* (Dearborn, 28-30 May 1997), IEEE Computer Society Press, Los Alamitos, 1997. Accessible at http://www.cc.gatech.edu/fac/Melody.Moore/papers/WPC97.ps

[15] A.R. Puerta, "The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development", *Proc. of 2nd Int. Workshop on Computer-Aided Design of User Interfaces CADUI'96* (Namur, 5-7 June 1996), Presses Universitaires de Namur, Namur, 1996, pp. 19-35.

[16] A.R. Puerta and J. Eisenstein, "Towards a General Computational Framework for Model-Based Interface Development Systems", *Proc. of ACM Conf. on Int. User Interfaces IUI'99* (Redondo Beach, 5-8 January 1999), ACM Press, New York, 1999, pp. 171-178.

[17] F. Ricca, P. Tonella, I.D. Baxter, "Restructuring Web applications via Transformation Rules", *Proc.of IEEE Workshop on Source Code Analysis and Manipulation SCAM'2001* (Florence, 5-9 November 2001), IEEE Computer Soc. Press, Los Alamitos, 2001, pp. 150-160. Accessible at http://star.itc.it/tonella/papers/scam2001. ps

[18] E. Stroulia, J. Thomson, Q. Situ, "Constructing XML-speaking wrappers for WEB Applications: Towards an Interoperating WEB", *Proc. of IEEE 7th Working Conference on Reverse Engineering WCRE'2000* (Brisbane, 23-25 November 2000), IEEE Computer Society Press, Los Alamitos, 2000, pp. 59-69.

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

[20] D. Thevenin, "Adaptation en Interaction Homme-Machine: le cas de la Plasticité", *Ph.D. thesis*, Grenoble, France, 2001. Accessible at http://iihm.imag.fr/publs/2001/

[21] J. Vanderdonckt and F. Bodart, "Encapsulating Knowledge for Intelligent Interaction Objects Selection", *Proc. of ACM Conf. on Human Aspects in Computing Systems InterCHI'93* (Amsterdam, 24-29 April 1993), ACM Press, New York, 1993, pp. 424-429. Accessible at http://www.isys.ucl.ac.be/bchi/members/jva/pub/Vanderdonckt-InterCHI93.pdf

[22] J. Vanderdonckt, L. Bouillon, N. Souchon, "Flexible Reverse Engineering of Web Pages with VAQUITA", *Proc. of IEEE 8th Working Conference on Reverse Engineering WCRE'2001* (Stuttgart, 2-5 October 2001), IEEE Computer Society Press, Los Alamitos, 2001, pp. 241-248

[23] Available on http://perso.wanadoo.fr/ablavier/TidyCom/).

[24] Available at http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/001/596/msdncompositedoc.xml.)

[25] The XIML Consortium, 2002, http://www.ximl.org

[26] Tidy accessible at http://www.w3.org/People/Raggett/tidy