Adapting UsiXML User Interfaces to Cultural Background

Iyad Khaddam
Consulting Company for Computers
& Communications (4C) - Consultant
Damascus, Syria
+963 944 266 475
iyadkh@gmail.com

Jean Vanderdonckt
Louvain School of Management, UCL
Place des Doyens, 1
B-1348 Louvain- la-Neuve (Belgium)
+32 10 478525
jean.vanderdonckt@uclouvain.be

ABSTRACT
Adapting a user interface to the end user's cultural background today remains an open challenge since many underlying issues are not yet solved. This paper addresses this challenge by reporting on a selected series of these issues, by structuring them according to Nielsen's linguistic model of interaction, and by discussing how each issue can be supported by incorporating its solution into a User Interface Description Language, such as User Interface eXtensible Markup Language (UsiXML), at the level of a Concrete User Interface (CUI). In particular, the problem of right-to-left (RTL) versus left-to-right languages (LTR) languages is discussed through a series of adaptations of algorithms and techniques that support the automated generation of Arabain graphical user interfaces based on UsiXML.

Categories and Subject Descriptors
H.1.2 [**Information Systems**]: Models and Principles – *User/Machine Systems*. H5.2 [**Information interfaces and presentation**]: User Interfaces – *Prototyping; user-centered design; user interface management systems (UIMS)*.

General Terms
Design.

Author Keywords
Arabization, Cultural background, Globalisation, Localisation, LTR, RTL, UsiXML.

INTRODUCTION
Right To Left (RTL) languages are languages that are written from Right to Left, like: Arabic, Farsi, Urdu and Hebrew. They use a different set of letters than Latin Counterparts. Similarly, some languages are read from bottom to top, as opposed to Latin languages that are read from top to bottom. We hereby define a RTL UI as a Graphical User Interface that accommodates the requirements posed by languages that are read from right to left. A RTL UI is the UI that meets the RTL language demands. RTL UI is an underestimated concept in the HCI community. At first glance, it may look like support for another set of languages, and sometimes it may be assumed as a localization of the product UI. Designing a RTL UI cannot be assimilated to designing a LTR UI and mirroring it by symmetry in order to obtain a corresponding RTL UI. Indeed, reading paths are different, location of widgets cannot be simply mirrored because of labels lengths, explanation and character sets. Existing algorithms for UI automated layout do not consider these aspects at all.

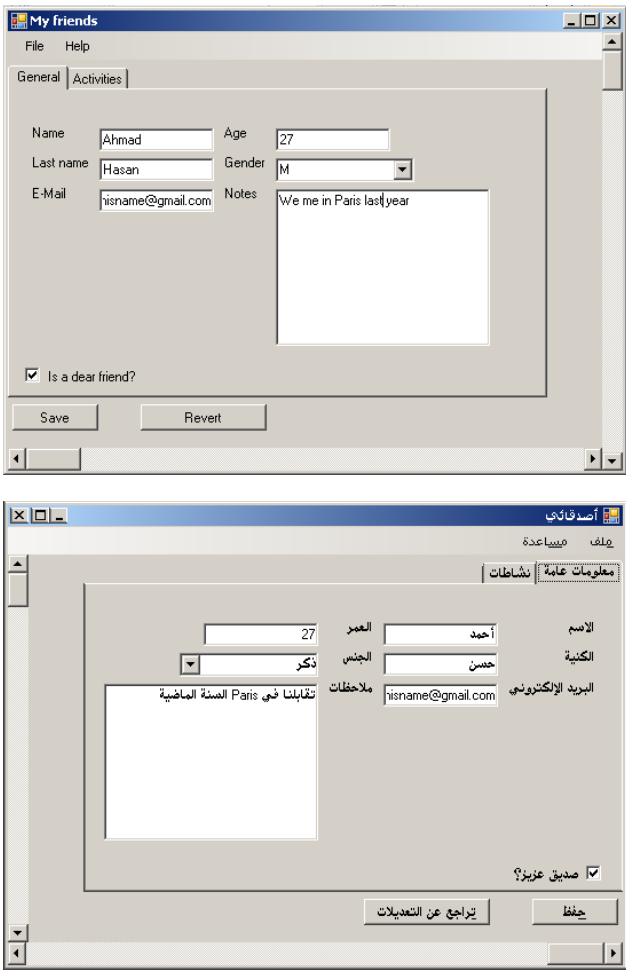
RTL UI has two aspects that should be treated together: the localization and the orientation (mirroring). If we fail to address one of them, the resulting UI will not be acceptable to users who are RTL language speakers. UI *localization* is addressed in multiple works, like in [20]. RTL is a common property among a set of written languages. This implies that each of these languages needs to be localized separately. Another less common property among the RTL languages is that most of them use the Arabic alphabet but add some extended letters (Hebrew has its own alphabet). Arabic can represent all the RTL languages as it has many features which do not exist in the rest of the RTL languages. Therefore, support for Arabic will enable support for the rest of the RTL languages that are similar in principle.

Figure 1 shows the characteristics of RTL UI versus LTR UI. We can easily notice the mirroring effect. The form header is mirrored, and components order is reversed The Close, Maximize, Minimize, Caption and Icon controls are all rearranged from right to left. The same effect applies to the menu and the tabs. Inside the tab, we can see that the table also is mirrored. The first column (that contains the labels: Name, Last name …) is positioned to the right. Table columns now flow from right to left which is the mirror of the LTR table layout. The vertical scroll is on the right, “Save” and “Revert” buttons positions are mirrored.

RTL affects controls too. We note the change of writing direction inside the text boxes. The combo box rendering is also mirrored (the “Gender”), the drop down image is positioned to the right of the combo box in the RTL version. The check box caption is also mirrored. Menus are also affected. In Figure 2, we show the effect of RTL on menus and sub menus, they expand to the right in the RTL version. The triangle image before the menu item “Transform…” is mirrored, and also is the arrow after.

An interesting difference to note is the horizontal bar. Although a horizontal bar is horizontally symmetrical, but the scroll bar starts from the left in LTR UI and from the right in the RTL UI. This is important to note in case we shrink a RTL window; the right view (controls on right) should always be visible and no need to scroll to the right.

Another interesting difference can be seen by looking at the “Notes” text box control. This control allows multiple lines, Pressing “Enter” key takes the cursor to the beginning of new line (\n + \r). While “beginning a new line” means in LTR “the left of the line”, RTL redefines this meaning as “the right of the line”. Writing direction switching may occur in a RTL text. The “Notes” field (in the RTL version) provides an example. The direction is switched using a combination of keyboard keys. In this example, the user wrote in Arabic, switched direction to LTR then back forth to RTL to continue his phrase. This can occur many times, and should not affect the readability of text.



Name: Ahmad
Last name: Hasan
E-Mail: hisname@gmail.com
Age: 27
Gender: M
Notes: We me in Paris last year
☑ Is a dear friend?

الاسم: أحمد
الكنية: حسن
البريد الإلكتروني: hisname@gmail.com
العمر: 27
الجنس: ذكر
ملاحظات: تقابلنا في Paris السنة الماضية
☑ صديق عزيز؟

Figure 1. a sample English UI and the localized RTL version (in Arabic Language)

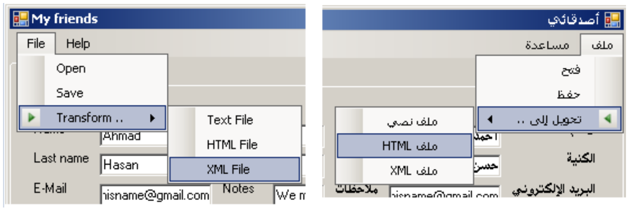


Figure 2. A sample LTR and the localized RTL version (in Arabic language).

“Tab” behavior is also affected with RTL (Tab to move focus to next enabled element). In the LTR version, pressing “tab” moves focus between controls in sequence: “Name”, “Last name”, “E-mail”, “Age”, “Gender” then “Notes”. The same behavior is expected in the RTL version. If the “Tab” key behavior is related to element positioning, it won’t work in the RTL version (unless positioning considers RTL). The last thing to notice is the shortcuts. As RTL languages use non-Latin alphabet, keyboard shortcuts will be different in both versions. The RTL version will use keyboard shortcuts from its own language alphabet. This problem is related to Localization more than to RTL.

From the above example, we summarize RTL effect on UI in 2 ways:

- The orientation (the mirroring)
- The localization

The TRL localization has special characteristics that don’t exist in Latin-alphabet languages, which makes the localization a challenge. We identify from the examples before the following characteristics for RTL localization:

- Text localization
    - Text localization: language encoding and character set (alphabet).
    - Direction switching: direction of text writing.
- Graphics localization
    - RTL sensitive graphics (non-horizontally symmetrical)
    - Images with text inside
    - Other localizable images (country flag…)
- Control localization
    - Control rendering: ex: label control should support writing from right to left.
    - Control behavior: controls should be aware of special behavior for special keys (like pressing “enter” key in a text area)

In the following sections, we discuss the support provided by the current version of UsiXML and explain our contribution to provide a full support for RTL.

RELATED WORK

As we mentioned before, the RTL issue is underestimated in the CHI research area. While localization is discussed in many works, rare effort was spent on RTL languages localization.

In the market, we can find well-arabized products and most of them are built on Microsoft Windows Operating System (As it was the pioneer in providing support for RTL languages) or on the web. ERP products were forced to provide RTL support due to market pressure. Hau in [7] shows the awareness of the RTL issue in the ERP industry.

Writers [5] explain the Arabic language characteristics and explain the challenges of the language in the context of providing OS support for Arabic. They discuss the issues related to encoding, character shaping and the “cursive” or “handwritten” style of writing in Arabic, vowels, numbers shapes and the mirroring effect on visual screens.

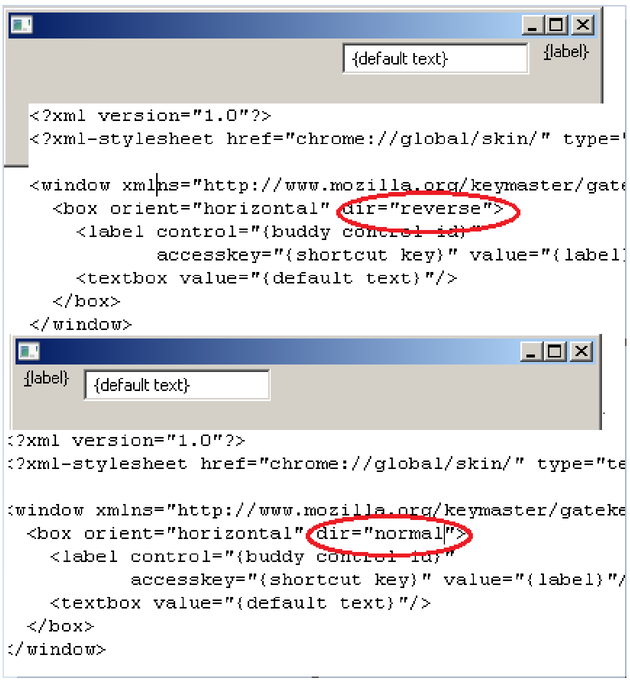
Rejmer et al. [20] discussed internationalization for a product as a case study. They came up with a set of guidelines to help internationalization/localization of a product. Their work handled the case of western languages (Latin alphabet) and didn’t discuss the case of RTL nor non-Latin alphabet languages. Their guidelines need reviewing to be adapted for RTL languages.

Other UI languages address the RTL in different ways. XUL [27] (XML User Interface Language) is the UI language used by Mozilla to create feature-rich cross platform applications. Firefox UI is built using this language.

XUL supports RTL UI by providing the “dir” property at the UI element which is the base of all elements. The “dir” property can have one of two values: normal, reverse. The “normal” means: “position elements in the container according to their order in the xml file”. The “reverse” value means: “position elements in the container according to their *reverse* order in the xml file”. Figure 3 shows the difference. XUL does not directly address the RTL concept, but their concept of “reverse” fixes the orientation issue. On the other side, it doesn’t address the control localization nor provide support to it. When the dir property is “reverse”, this doesn’t imply that the final control to be used is a control that supports RTL. XUL depends on the rendering framework to determine the final control (localized version). Thus it provides a localized version of Firefox for each language. The problem we note with XUL approach is that each localized version of the product will have a localized version of the design (the XUL file is copied for each language). This imposes a maintenance/update problem.

XAML is a markup system that underlies user interface components of Microsoft's .NET Framework 3.0 and above [26]. XAML supports RTL by adding a “FlowDirection” property to the containers and UI elements that takes one of the values: “LeftToRight”, “RightToLeft”. This causes the expected effect of RTL to be applied on the container and/or the element. Figure 4 gives an example.

The FlowDirection property is inherited by all the elements in the objects hierarchy; all the elements in a container inherit its container’s property. Thus, we only need to set this property on the window level and all the inside elements will inherit it. Note the effect on the combo box, it becomes RTL in response to the window FlowDirection property value.



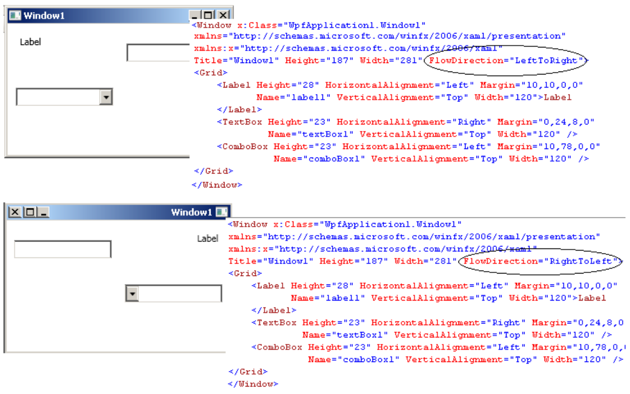
```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type='

<window xmlns="http://www.mozilla.org/keymaster/gate
  <box orient="horizontal" dir="reverse">
    <label control="{buddy control-id}"
           accesskey="{shortcut key}" value="{label}
    <textbox value="{default text}"/>
  </box>
</window>
```

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="te

<window xmlns="http://www.mozilla.org/keymaster/gateke
  <box orient="horizontal" dir="normal">
    <label control="{buddy control-id}"
           accesskey="{shortcut key}" value="{label}"/
    <textbox value="{default text}"/>
  </box>
</window>
```

Figure 3. XUL property dir and effect of the 2 different values: normal and reverse.



```
<Window x:Class="WpfApplication1.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="Window1" Height="187" Width="281" FlowDirection="LeftToRight">
<Grid>
    <Label Height="28" HorizontalAlignment="Left" Margin="10,10,0,0"
           Name="label1" VerticalAlignment="Top" Width="120">Label
    </Label>
    <TextBox Height="23" HorizontalAlignment="Right" Margin="0,24,8,0"
             Name="textBox1" VerticalAlignment="Top" Width="120" />
    <ComboBox Height="23" HorizontalAlignment="Left" Margin="10,78,0,0"
              Name="comboBox1" VerticalAlignment="Top" Width="120" />
</Grid>
</Window>
```

```
<Window x:Class="WpfApplication1.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="Window1" Height="187" Width="281" FlowDirection="RightToLeft">
<Grid>
    <Label Height="28" HorizontalAlignment="Left" Margin="10,10,0,0"
           Name="label1" VerticalAlignment="Top" Width="120">Label
    </Label>
    <TextBox Height="23" HorizontalAlignment="Right" Margin="0,24,8,0
             Name="textBox1" VerticalAlignment="Top" Width="120" />
    <ComboBox Height="23" HorizontalAlignment="Left" Margin="10,78,0,
              Name="comboBox1" VerticalAlignment="Top" Width="120" />
</Grid>
</Window>
```

Figure 4. XAML property FlowDirection and effect of the 2 different values: LeftToRight and RightToLeft.

The FlowDirection property can be overridden on the element’s level, this enables the designer to give a certain element a different direction than the container. This can be helpful in UI where a mixed content of LTR and RTL languages are used.

The problem XAML faces is that when we have a set of RTL interfaces (windows), then each must be set individually. This is because each file defines a window only. The other problem is that we need to copy the design (the XAML file) to obtain the two UIs (RTL and LTR) as the FlowDirection property value is static.

CURRENT RTL SUPPORT IN UsiXML

Architectural overview

UsiXML is MDA-compliant. The UsiXML development process [23] aligns the UsiXML models with the MDA models as follows:

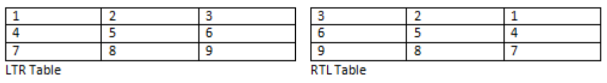
MDA	UsiXML model
Computing Independent Model (PIM)	Task, Domain
Platform-Independent Model (PIM)	Abstract User Interface (AUI)
Platform-Specific Model (PSM)	Concrete User Interface (CUI)
Code	Final User Interface (FUI)

Table 1. UsiXML compliance to MDA.

RTL is a platform-specific issue. This implies that supporting RTL in UsiXML should be considered in the CUI model. RTL should not affect the Task, Domain or AUI models to keep UsiXML MDA-compliant.

The orientation

Current version of UsiXML doesn't make any difference between LTR and RTL, and is assumed to be LTR. UsiXML supports element positioning and alignment (Right, Left and Center). Positioning is supported by the employment of 2dGraphicalContainer objects. Positioning is different from orientation. Figure 5 shows this difference on a table.



1	2	3
4	5	6
7	8	9

LTR Table

3	2	1
6	5	4
9	8	7

RTL Table

Figure 5. The difference between positioning and orientation.

RTL Localization in UsiXML

Localization is supported in UsiXML using the resource model, which provides limited support for RTL languages localization. The problems with current implementation of localization in UsiXML are as follows:

Text Translation

UsiXML supports text localization. The same technique is applicable for RTL languages. Anyway, the issue in RTL text translation is in the storage of local text inside the xml file. The encoding used to save the xml file needs to support the RTL language. Using Unicode can be a solution for this issue.

Direction switching

UsiXML does not sense the change of writing direction in a text. It relies on the input/output FUI objects to do the rendering. This is a concern in case the target platform doesn't support RTL, or if output FUI objects don't support RTL. Enabling writing direction-change sensing would provide information to transformers (from CUI to FUI) to enhance the generated UI in case FUI controls don't support RTL (for example: a transformer may render the RTL language text as an image instead of text).

Graphics Localization:

All the issues we mentioned before regarding graphics localization can be solved using the current UsiXML localization support. If we have an English UI resource file with *n* images, the localized UI resource for another LTR language (say French) will contain *n* images also but with *m* (m<=n) localised images (ex. flag images and images with text). In the case of localization for an RTL language, the number of localized images may be larger than *m*. This is due to the non-horizontally symmetrical images that may exist in the original design (ex. Horizontal arrows need to be RTL localized). We will address this issue in the next section.

Control Localization:

UsiXML does not support any kind of control RTL localization. All CUI controls are direction insensitive. This is also left for the FUI objects. The problems with leaving them to the FUI are:

1- The control's direction property is hidden in the transformer. Reverse engineering to CUI will fail to identify the control's direction.

2- No transformer can generate a UI that contains both LTR and RTL FUI controls (Figure 1 gives an example: the e-mail text box must be LTR in both versions).

In the next section, we will provide enhancements to UsiXML to overcome the shortage in the current version.

UPDATING UsiXML DESIGN TO SUPPORT RTL UI

UsiXML has a unique characteristic over other UI models, which is the employment of multiple models in one. UsiXML designers design the UI of a concept, while in other UI models, they focus on the design of a single interface (window). One of the important models in our case is the context model.

Updating UsiXML class design

RTL is a platform property, and the context model describes the platform. This direct mapping gives an intuitive solution to the problem: extend the Platform class to contain a new property: "dir". The "dir" is an optional property that can have one of two values: "LTR" (the default) or "RTL". The other update to the design is adding a new property to the CUI class: 2DgraphicalCio. This will support RTL in all graphical user interface elements (containers or individual components). The "dir" property for the 2DgraphicCIO is inherited from the containing container. Root container (the window) inherits from the platform "dir" property.

Using the "dir" property

From a designer perspective:

The designer creates a new context for the RTL UI, where he can localize the messages and other resources. He also sets the “dir” property of the platform to the value “RTL”. There is no need to set the “dir” property for containers and elements as the default value for them will be that in the context. Wherever there is an exception (i.e: the GUI is orientation-independent), a fixed value needs to be set for that gui element’s “dir” property. In Figure 1, we can see an example: the text box of the e-mail; e-mails are always written in English, so the text box should always be LTR.

From a transformer perspective

The transformer checks the “dir” property for each 2dgraphicalCIO. As we discussed before, the “dir” property is either context-dependent or static value. In both cases, it is resolved according to current employed context. Transformers have the choice now to render FUI correctly and to use the correct individual controls that support the CIO orientation.

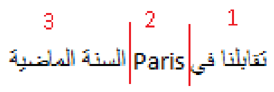
Optional design modifications

In the section before, we mentioned the issue of non-horizontally symmetrical images. Although this issue doesn’t affect support for RTL in UsiXML (as support is provided using resource files), but an enhancement to reduce the number of images in the RTL UI localized resource may save designer’s time. Image rotation can be achieved programmatically. Transformers can produce the RTL version of these images when applied. All we need to do is to denote the images that are not horizontally symmetrical. We suggest adding a new property to the imageComponent clas; the “ImageDir” property. This property accepts one of the values: “RTL”, “LTR”, “*empty*”. If the physical image is for the LTR UI, then the value is LTR and vice versa. The transformer decision to rotate the image is explained in Table 2.

		ImageDir		
		LTR	RTL	*empty*
dir	LTR		Rotate	
	RTL	Rotate		

Table 2. The decision table used by transformers to rotate the image is based on the two properties of the Image.

The rotated version of the image can be saved and handled by the transformer internally and the transformation process is re-entrant. Direction switching can be handled by adding a direction tag separator when a switching occurs. We can use two tags: <LTR> and <RTL>. A string like:



Can be rewritten to represent direction switching as follows:

<RTL>تقابلنا في</RTL><LTR>Paris</LTR><RTL> السنة الماضية</RTL>

This way, a transformer can have a choice to handle cases where an FUI control doesn’t support RTL to be rendered correctly. A valid approach to render the above text appropriately is to generate 3 labels in that the left-most will contain the 3rd part of the string and so on. In the case where the OS doesn’t support the alphabet, a transformer can produce an image of the above text. If RTL is well supported in a platform, the transformer can ignore the direction tags.

RTL AND USABILITY

Many resources can be found regarding usability tips and practices. In this paper, we present some of them and show the effect of RTL on known usability (RTL usability) tops and practices. We do not intend to provide a full guide, but just to give a sense of the differences. Some useful usability tips exist : http://www.ambysoft.com/essays/user-InterfaceDesign.html. We excerpt two of them to discuss.

Tip 1: **Align fields effectively:** When a screen has more than one editing field, you want to organize the fields in a way that is both visually appealing and efficient. I have always found the best way to do so is to left-justify edit fields: in other words, make the left-hand side of each edit field line up in a straight line, one over the other. The corresponding labels should be right-justified and placed immediately beside the field. This is a clean and efficient way to organize the fields on a screen.

Our comment: This is an LTR thinking. To make this tip applicable in RTL UI, we need to replace the left words with right and vice versa.

Tips 2: ***Justify data appropriately:*** *For columns of data, common practice is to right-justify integers, decimal align floating-point numbers, and to left-justify strings.*

Our comment: this is a cultural preference. In Arabic, the more preferred way is to center-align numbers and right-justify strings.

TOWARDS CULTURALLY-AWARE UsiXML UIs

The last examples suggest that transforming UsiXML-based UIs for Arabic languages and culture is certainly a matter of revisiting existing techniques for selecting and placing widgets in a GUI. In this section, we briefly consider two techniques used for automating the production of GUIs: one technique that automatically select widgets depending on domain parameters and one technique for automated layout of these widgets.

Automatic selection of widgets

[14] provides a decision-tree based technique in order to automatically select widgets (e.g., check boxes, radio buttons, list boxes, combination boxes) based on parameters coming from the domain model (e.g., data type, number of possible values, number of values to choose). While the cognitive principles and usability guidelines that have been considered in order to build this technique remain universal in principle, there are still some adjustments that are not considered for Arabian languages because the technique assumes that the GUI is based on Western properties that are not necessarily applicable or valid for non-Western countries and cultures. For instance, Table 3 depicts some

adjustments that have been introduced in order to take into account RTL languages: the rightmost columns indicates the Concrete Interaction Object (CIO) to be selected depending on the following parameters from the domain model: for a simple choice in a known domain, number of secundary values (Nsv), expandable domain (Exp), Continuous domain (Cont), Number of possible values (Npo), precision, and orientation. Most of the time, the CIOs remain the same, but their right alignment is preferred and the combination of widgets is arranged so that the reading order is correct.

Automatic laying out of widgets

[15] describes an algorithm that automatically positions selected widgets resulting from the previous step in a container according to a technique called "Right-Bottom". This dynamic strategy is explained to be more flexible than a static strategy [16]. Let Si denotes the CIO placed at time i. Si+1 is the next CIO to be placed. The idea of the right/bottom strategy consists of following the visual continuity principle by either placing Si+1 on the right of Si or beneath Si. The idea is to perpetuate this layout technique until all selected widgets have been placed. The placement strategy is defined as the following:

```
if the total length does not exceed the limit
then
   place $S_{i+1}$ with horizontal sequencing
   three cases are to be considered
   1. height ($S_i$) = height ($S_{i+1}$)
      apply proportional uniformity
   2. height ($S_i$) > height ($S_{i+1}$)
      if $S_{i+1}$ = edit box
         then
            if $S_i$ = list box or edit box
               then apply bottom justification
               else apply upper justification
   3. height ($S_i$) < height ($S_{i+1}$)
      if available space is sufficient
      then apply bottom justification
      else maximize upper justification
else
   place $S_{i+1}$ with vertical sequencing.
```

Nsv	Exp	Cont	Npo	Precision	Orientation	CIO
> 0						list box with right aligned items
= 0	yes					combination box with right aligned edit field
	no	no	[2,3]			radio-button with Npo items
			[4,7]			radio-button with Npo items + group box
			[8,Tm]			list box with right aligned items
			[Tm+1,2Tm]			scrolling list box with right aligned items
			> 2Tm			scrolling drop-down list box with right aligned boxes
		yes	[1,10]	low	vertical	scroll bar
					horizontal	Scale
					circular	pie diagram in counterclockwise presentation
					undefined	scale
				high	vertical	vertical thermometer
					horizontal	horizontal thermometer
					circular	dial with Arabian presentation
					undefined	horizontal thermometer
			[11,Tm]	high		spin button
				low		scale
			> Tm	high		spin button
				low	vertical	scroll bar
					horizontal	scale
					circular	dial with Arabian presentation
					undefined	scale

Table 3. Rules for selecting Concrete Interaction Objects (Adapted from [14]).

If we want to adapt this technique, we must revert most of mathematical relationships that are logically defined. This then becomes the Left-Bottom strategy:

```
if the total length does not exceed the limit
then
    place $S_{i+1}$ with horizontal sequencing from left to right
    three cases are to be considered
    1. height ($S_i$) = height ($S_{i+1}$)
         apply proportional uniformity
    2. height ($S_i$) > height ($S_{i+1}$)
         if $S_{i+1}$ = edit box
         then
            if $S_i$ = list box or edit box
               then apply upper justification
               else apply bottom justification
    3. height ($S_i$) < height ($S_{i+1}$)
         if available space is sufficient
         then apply upper justification
         else maximize bottom justification
else
    place $S_{i+1}$ with vertical sequencing.
```

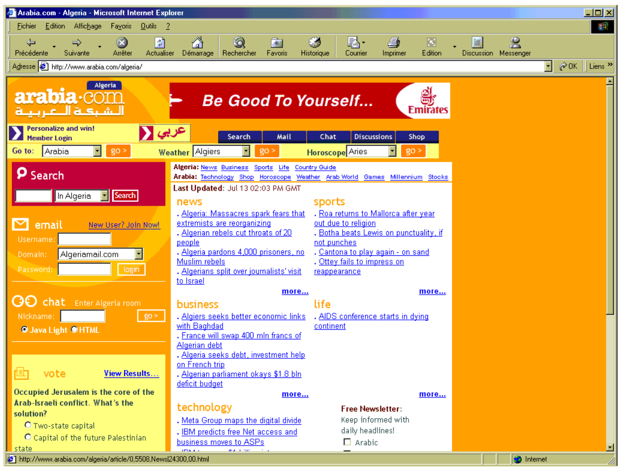
	PDI		IDV		MAS		UAI		LTO	
	rank	score	rank	score	rank	score	rank	score	rank	score
Arab Countries	7	80	26/27	38	23	53	27	68		
Argentina	35/36	49	22/23	46	20/21	56	10/15	86		
Australia	41	36	2	90	16	61	37	51	15	31
Austria	53	11	18	55	2	79	24/25	70		
Bangladesh									11	40
Belgium	20	65	8	75	22	54	5/6	94		

Table 4. Some values of the Five Cultural Dimensions as estimated in [10].

At first glance, at are these little adjustments may appear sufficient. But at second glance, updating a GUI for Arabian languages is more than a simple matter of repositioning widgets. Indeed, culture has been identified [7,8,9,13] has a very important aspects that is often forgotten in the design of globalized user interfaces and localized user interfaces, thus clearly affecting the identity of the web site in the country [12]. Marcus [10,11] classified 53 countries according to Hofstede [8] five cultural dimensions: PDI: Power distance index, IDV: Individualism index, MAS: Masculinity index, UAI: Uncertainty avoidance index, and LTO: Long-term orientation index. For instance, Table 4 reproduces some lines of this work (Source: http://www.amanda.com/resources/hfweb2000/hfweb00.marcus.html).

While some of the indexes are mainly indicative and not necessarily true in every circumstance, they suggest aspects to be considered in culturally-aware UIs. For instance, figure 6 depicts the same web site but with complete redrawing while keeping the same contents. While consistency across languages may certainly be desirable, other aspects could be considered as well. For instance,

Marcus suggests that the following countries have very different masculinity index: 95 Japan, 79 Austria, 63 South Africa, 62 USA, 53 Arab countries, 47 Israel, 43 France, 39 South Korea, 05 Sweden. This is not at all reflected in the GUI design.


Arabia.com - Algeria - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Suivante Arrêter Actualiser Démarrage Rechercher Favoris Historique Courier Imprimer Edition Discussion Messenger
Adresse http://www.arabia.com/algeria/
Algeria
arabia.com
الشبكة العربية
Be Good To Yourself...
Emirates
Personalize and win!
Member Login
عربي
Search Mail Chat Discussions Shop
Go to: Arabia go >
Weather Algiers go >
Horoscope Aries go >
Search
In Algeria Search
email New User? Join Now!
Username:
Domain: Algeriamail.com
Password: login
chat Enter Algeria room
Nickname: go >
Java Light HTML
vote View Results...
Algeria: News Business Sports Life Country Guide
Arabia: Technology Shop Horoscope Weather Arab World Games Millennium Stocks
news
sports
business
life
technology
more...
Free Newsletter:
Arabic
http://www.arabia.com/algeria/article/0,5508,News|24300,00.html

Last Updated: Jul 13 02:03 PM GMT

news
. Algeria: Massacres spark fears that extremists are reorganizing
. Algerian rebels cut throats of 20 people
. Algeria pardons 4,000 prisoners, no Muslim rebels
. Algerians split over journalists' visit to Israel

sports
. Roa returns to Mallorca after year out due to religion
. Botha beats Lewis on punctuality, if not punches
. Cantona to play again - on sand
. Ottey fails to impress on reappearance

business
. Algiers seeks better economic links with Baghdad
. France will swap 400 mln francs of Algerian debt
. Algeria seeks debt, investment help on French trip
. Algerian parliament okays $1.8 bln deficit budget

life
. AIDS conference starts in dying continent

technology
. Meta Group maps the digital divide
. IBM predicts free Net access and business moves to ASPs

Occupied Jerusalem is the core of the Arab-Israeli conflict. What's the solution?
○ Two-state capital
○ Capital of the future Palestinian state

Free Newsletter: Keep informed with daily headlines!

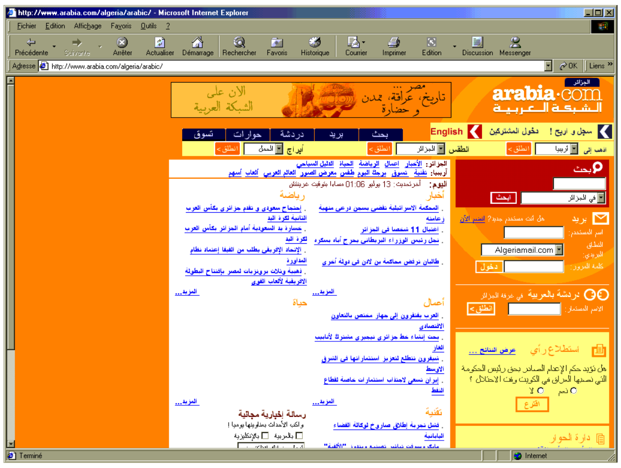

http://www.arabia.com/algeria/arabic/ - Microsoft Internet Explorer
arabia.com
English
Algeriamail.com

Figure 6. The same web site in English (LTR) and in Arabic (RTL).

A CLASSIFICATION OF ISSUES FOR ARABIAN UIS

In order to classify a manual operation and, therefore, a subsequent beautification operation, Nielsen's linguistic model of interaction [14] was selected for these reasons: it decomposes a human-computer interaction in terms of seven inter-related, but independent, levels with a communication protocol between them; it has already been successfully used to classify usability guidelines according to their level of importance; and it allows identification criteria to univocally locate each modification to one and only one level. Table decomposes a simple goal (i.e., delete a paragraph in a letter) into subsequent units of interaction for each level:

- **Level 1 (Goal)**: expresses a user's mental goal, such as "search for a particular customer having a water meter in a specific region".
- **Level 2 (Pragmatic)**: translates this mental goal into a task to be carried out in the system according to the system concepts, such as "search for a subscriber having at least one water meter in zone *x*".
- **Level 3 (Semantic)**: translates the real-world objects into system objects and functions, such as "search for a subscriber with a code region filled in".
- **Level 4 (Syntactic)**: structures the semantic into an ordered sequence of operations in time and space, such as "select a zone code from the list and launch a query".
- **Level 5 (Lexical)**: decomposes each operation into the smallest possible pieces of information, such as "a zone code".
- **Level 6 (Alphabetic)**: specifies the unit of information (e.g., a lexeme, a metric) for each information item, such as "an integer for representing the zone code".
- **Level 7 (Physical)**: specifies the physically-coded information in terms of light, sound, color, etc., such as "display the integer in black on white for input".

Level	Title	Units	Definition	Example	World
1	Goal	Concepts of real world	Mentalization of a goal, a wish in the user's head	Delete a paragraph from my letter	Conceptual
2	Pragmatic	Concepts of system	Translation of a goal into system concepts	Delete 6 lines of the current paragraph in the edited text	
3	Semantic	Detailed functions	Real world objects translated into system objects manipulated by functions	Delete a certain amount of lines	
4	Syntactic	System sentences	Time & space sequencing of information units	DELETE 6	Perceptual
5	Lexical	Information units	Smallest elements transporting significant information: word, figure, screen coordinates, icon	[DELETE] command, [6] number	
6	Alphabetic	Lexems	Primitive symbols: letter, numbers, columns, lines, dots, phonems, …	D, E, L, E, T, E, 6	Physical
7	Physical	Physically coded information	Light, sound, physical moving	Pressing [CTRL]+[D] followed by [6]	

Table 5. Definition of the seven levels of Nielsen's linguistic model of interaction [14].

According to Nielsen's linguistic model, we can classify the problems discussed in this paper as follows:

Text Localization is the replacement of strings with the Arabic opponents; it replaces primitive symbols, so it fits at level 6 (alphabet). Writing direction switching involves syntactical handing of the text, so it fits at level 4.

Image Localization is just like Text Localization if we see localized images as the path to the image (the unit of information is the image path). If we look at images as the unit of information (like when generating images dynamically), then Image Localization is considered at level 5 (Lexical).

Control Localization fits at level 5 (lexical) or at level 4 (Syntactic) according to the way we implement it. We can use "RTLLabel" and "LTRLabel" instead of "Label" to implement localized controls, then we are adding new lexemes and localization fits at level 5. If no new lexemes are introduced, we can still support localization by introducing a new syntax: *Label rtl, "any Arabic string"* and the Control localization fits at level 4. We believe that UsiXML should implement control localization at level 4.

The Orientation (RTL layout) is at level 4 since it is related to spatial aspects, but it can be seen at level 3 (semantic). At level 4, the UI will be described as:

```
Add Window RTL
Add Label RTL, "An Arabic string"
Add TextBox RTL
…
Add Label LTR, "E-Mail"
Add TextBox LTR
```

While at level 3, we can describe the same UI above as:

```
UIContext RTL
Add Window
Add Label "An Arabic string"
Add TextBox
…
Add Label LTR, "E-Mail"
Add TextBox LTR
```

In this example, the page direction is defined globally and controls inherit the orientation property. The direction can be still set manually for a specific control if different than the global setting. The algorithms we discussed before affect widget selection and layout, thus they fit at level 4. We note also that a little semantic is used which is related to the orientation problem.

The cultural differences and effect on Arabian UI (the Hofstede's cultural dimensions) are at level 2 (Pragmatic) since taking into account cultural aspects is very much based on high-level considerations.

CONCLUSION

In this paper, we explained the challenges imposed by RTL languages. We provided a solution in UsiXML to handle all the challenges we mentioned. The proposed solution provides better support for designers than other UI models thanks to unique characteristics of UsiXML. The "dir" property of the platform is used to handle the orientation problem and the "dir" property of the ui elements provides support for localized controls. Direction switching can be fixed using the direction tag separators. Image localization for non-horizontally symmetrical ones is also handled. On top of all that, the designer maintains only one copy of the design thanks to the context model in UsiXML.

FUTURE WORK

The Orientation problem can be generalized to address Top To Bottom UI and Bottom To Top UI. Studying the effect of RTL on 3DUI can be another interesting area to evaluate. In the context of cultural differences effect on UI, more work should be spent on Arabian UI to come up with concrete results that may help to enhance Arabian usability guidelines.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the FP7 Serenoa project funded by the European Commission and the UsiXML project (ITEA2 Call3) under reference 2008026.

REFERENCES

1. Bodart, F., Hennebert, A.-M., Leheureux, J.-M., and Vanderdonckt, J. Towards a Dynamic Strategy for Computer-Aided Visual Placement. In *Proc. of 2nd ACM Workshop on Advanced Visual Interfaces AVI'94* (Bari, 1-4 June 1994). T. Catarci, M.F. Costabile, S. Levialdi, G. Santucci (eds.), ACM Press, New York, 1994, pp. 78-87.
2. Bodart, F., Vanderdonckt, J. On the Problem of Selecting Interaction Objects. In *Proc. of BCS Conf. HCI'94 "People and Computers IX"* (Glasgow, 23-26 August 1994), Cambridge University Press, Cambridge, 1994, pp. 163-178.
3. Coyette, A. and Vanderdonckt, J. A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces. In *Proc. of 10th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'2005* (Rome, 12-16 September 2005). Lecture Notes in Computer Science, Vol. 3585, Springer-Verlag, Berlin, 2005, pp. 550-564.
4. Cyr, D., Head, M., and Larios, H. Colour appeal in website design within and across cultures: A multi-method evaluation. International Journal of Human-Computer Studies, Vol. 68, No. 1-2, January 2010, pp.1-21.
5. DelGaldo, E.M. and Nielsen, J. International User Interfaces. John Wiley & Sons, New York, 1996.
6. Evers, V. and Day, D. The role of culture in interface acceptance. In *Proc. of the IFIP TC13 Int. Conf. on Human-Computer Interaction INTERACT'97* (Sydney, 14- 18 July 1997), Chapman & Hall, London, 1997, pp. 260-267.
7. Hau, E. and Aparicio, M. Software Internationalization and Localization in Web Based ERP. In *Proc. of the 26th Annual ACM Int. Conf. on Design of communication SIGDOC'2008* (Lisbon, 22–24 September 2008). ACM Press, New York, 2008, pp. 175-180.
8. Hertzum , M., Clemmensen, T., Hornbæk, K., Kumar, J., Shi, Q., and Yammiyavar, P. Usability constructs: a cross-cultural study of how users and developers experience their use of information systems. In *Proc. of the 2nd Int. Conf. on Usability and Internationalization* (Beijing, 22-27 July 2007). Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2007, pp. 317-326.
9. Hofstede, G. Culture and organisation: Software of the Mind. McGraw Hill, New York, 1997.
10. Hofstede, G. Culture's Consequences: International Differences in Work-Related Values. Sage Publications, Beverly Hills, 1980.

11. Mahemoff, M. and Johnston, L. "Culturally-aware" requirements for internationalized software. In *Proc. of 3rd Australian Conf. on Requirements Engineering ACRE'98* (Geelong, Australia, 26-27 October 1998), D. Fowler, L. Dawson (eds.), Deakin University, Geelong 1998, pp. 83-90. Accessible at http://mahemoff.-com/ paper/reqsi18n//

12. Marcus, A. and West Gould, E. Cultural Dimensions and Global Web User-Interface Design. *Interactions*, Vol. 7, No. 4, July 2000, pp. 32-46.

13. Marcus, A., Armitage, J., Frank, V., and Guttman, E. Globalization of User-Interface Design for the web. In *Proc. of 5th Int. Conf. on Human Factors and the Web* HFWeb'99 (Gaithersburg, 3 July 1999), 1999. Accessible at http://www.amanda.com/resources/HFWEB-99/HFWEB99.Marcus.html.

14. Nielsen, J. A Virtual Protocol Model for Computer-Human Interaction. *International Journal of Man-Machine Studies*, Vol. 24, No. 3, 1986, pp. 301–312.

15. Nomura, S., Ishida, T., Masaki, S., Yokozawa, M., and Shinohara, T. International Comparative Study of Identity as Presented on the Internet. In *Proc. of 6th Conf. on Human Factors and the Web HFWeb'2000* (Austin, 19 June 2000), Ph. Kortum, E. Kudzinger (eds.), University of Texas, Austin, 2000.

16. Patrick Rau, P.-L., Gao, Q., and Max Liang, S.-F. Good computing systems for everyone-how on earth? Cultural aspects. *Behaviour & Information Technology*, Vol. 27, No.4, July 2008, pp. 287-292.

17. Pérez-Quiñones, M.A., Padilla-Falto, O.I., and McDevitt, K. Automatic language translation for user interfaces. In *Proc. of the Int. Conf. on Diversity in computing TAPIA'2005* (Albuquerque, 19-22 October 2005). ACM Press, New York, 2005, pp. 60-63.

18. Portaneri, F. and Amara, F. Arabization of Graphical User Interfaces. In "International User Interfaces", E. Del Galdo, J. Nielsen (eds.), John Wiley & Sons, New York, 1996. http://www.langbox.com/staff/arastub.-html.

19. Rehm, M., Bee, N., Endrass, B., Wissner, M., and André, E. Too close for comfort?: adapting to the user's cultural background. In *Proc. of the Int. Workshop on Human-centered multimedia HCP'07* (Augsburg, 28-28 Sept. 2007). ACM Press, New York, 2007, pp. 85-94.

20. Rejmer, P., Cooper, M., and Vanderdonckt, J. Lessons Learned From Internationalizing a Web Site Accessibility Evaluator. In *Proc. of 3rd Int. Workshop on Internationalisation of Products and Systems IWIPS'2001 " Designing for global markets 3 "* (Milton Keynes, 11-14 July 2001), D.L Day & L.M. Dunckley (eds.). Digital Printing Services, The Open University, Milton Keynes, 2001, pp. 61-79.

21. Taylor, D. Global software: Developing applications for the international market. Springer-Verlag, Berlin, 1992.

22. Trompenaars, F. and Hampden-Turner, Ch. Riding the waves of culture, McGraw-Hill, New York, 1998.

23. UsiXML User Interface eXtensible Markup Language :http://www.w3.org/2005/Incubator/model-based-ui/ wiki/UsiXML

24. Vanderdonckt, J. Knowledge-Based Systems for Automated User Interface Generation: the Trident Experience. In *Proc. of CHI'95 Workshop on Knowledge-Based Support for the User Interface Design Process* (Denver, 7-8 May 1995). ACM Press, New York, 1995, pp. 21-33.

25. Vanderdonckt, J., Ouedraogo, M., and Yguietengar, B. A Comparison of Placement Strategies for Effective Visual Design. In *Proc. of BCS Conf. HCI'94 "People and Computers IX"* (Glasgow, 23-26 August 1994). G. Cockton, S.W. Draper, G.R.S. Weir (eds.), Cambridge University Press, Cambridge, 1994, pp. 125-143.

26. Wikipedia, *List of User interface Markup languages*. 2010. Accessible at http://en.wikipedia.org/wiki/List_of_user_interface_markup_languages

27. XUL,accessible https://developer.mozilla.org/En/XUL

28. Zahir, S., Dobin, B., and Gordon Hunter, M. Analysis of the cross-cultural dimensions of national web portals, Managing globally with information technology. Idea Group Publishing, Hershey, 2003