# Generating adaptable user interfaces using rich internet application

SABO COSMIN
Department Of Mathematics and Computer Science
North University of Baia Mare
Baia-Mare, 430122 Str. Dr. Victor Babeş Nr 62A
ROMÂNIA
cosmin_sabo@scream.ro http://cosmin.ubm.ro

NICOLAE TOMAI
Faculty of Economics and Business Administration
Babes-Bolyai University of Cluj Napoca
Cluj Napoca, 400591 Str. Teodor Mihali, Nr. 58-60
ROMÂNIA
nicolae.tomai@econ.ubbcluj.ro http://www.econ.ubbcluj.ro/~nicolae.tomai/

*Abstract:* - Personalization of interfaces by each user using user-friendly forms is a key concept to ensure interface accessibility. In this direction, we are using Extensible Markup Language (XML) as data source to generate interfaces. All users can adapt interface by modifying each component style, properties and events and modifications are saved as XML content. This user interfaces can be run as desktop application or in browser. This result in the generation of personalized multimodal user interfaces can be useful for many kinds of applications.

*Key-Words:* - Graphical User Interface; Adaptable User Interfaces, Rich Internet Application

## 1 Introduction

Personalization of interfaces by each user using user-friendly forms is a key concept to ensure interface accessibility. In this direction, we are using Extensible Markup Language (XML) as data source to generate interfaces. All users can adapt interface by modifying each component style, properties and events and modifications are saved as XML content. This user interfaces can be run as desktop application or in browser. This result in the generation of personalized multimodal user interfaces can be useful for many kinds of applications.

This concept is not design for a particular user or a particular application. The system is design to be used by class of users and set of tasks. Individual users can redesign interface and make the user needs more independent of the designer, and does not force the designer to decide about user specific needs.

Main goal of this system is to fit the interface to a specific user and to a specific task not only to the design phase specifications. Flexibility can be obtained using adaptable or adaptive interfaces.

An interface is called adaptive if it changes its own characteristics depending on the way the user interacts with the interface. Adaptive interfaces are a promising attempt to overcome contemporary problems due to the increasing complexity of human-computer interaction. They are designed to tailor a system's interactive behavior with consideration of both individual needs of human users and altering conditions within an application environment. The broader approach of intelligent user interfaces includes adaptive characteristics as a major source of its intelligent behavior.

A different way of helping a person to use a system more effectively is to adapt the user interface so that it fits better with our way of working with the system. Interface elements that have been adapted in this way include menus, icons, and the system's processing of signals from input devices such as keyboards [1]. An interface is called adaptable if it provides tools that make possible to change interface

characteristics. This tool gives the control over the adaptation to the user.
Adaptive interfaces can exhibit some unpleasant side effects such as surprising the user by moving or removing menu entries. Previous studies have also shown a desire for the user to be able to control and override the automatic system whenever needed [5].
My approach it is to provide exhaustive adaptation possibilities for all interface features of the interface.
The need for adaptive and personalized Rich Internet application puts a new dimension to adaptable interfaces. Instead of computing the adaptation steps at the server, Rich Internet Applications can use client-side approach and react immediately on user input. The focus of this paper is the conceptual introduction of client adaptable interfaces using rich internet application that a using XML files to generate the application interface and events. This client interface directly executes all necessary adaptation and save the modifications on the server. To customize client interface we don't need to reload pages like in the classic internet application or to user server side computing.

# 2 Problem Formulation

Main goal is to create a user interface adapter that can be accessed through any browser that has support for RIA applications or desktop applications must assure setting portability.
Each authenticated user should be able to have a different interface for other users, both in design and as functionality. Regardless of where the access the user interfaces, you can benefit from the changes made previously.
Based on this, result the following goals:
System independence: we do not impose a particular operating system for user.
Browser independence: the application should run on a wide variety of popular browsers.
User independence: by providing custom themes and skins.

## 2.1 Themes and skins features

Motivation to develop such a user interface is the user need to customize themes and skins.
In computing, skins may be associated with themes as custom graphical appearances (GUIs) that can be applied to certain software and websites to suit the tastes of different users. Software which is capable of having a skin applied is referred to as being skinnable, and the process of writing or applying such a skin is known as skinning. Applying a skin changes a piece of interface look and feel some skins merely make the interface more aesthetically pleasing, but others can rearrange elements of the interface, potentially making the program easier to use. Although often used simply as a synonym for skin, the term theme normally refers to less-complex customizations.
Another motivation is that almost all the users prefer to have the control of the application instead of requesting all the modifications needed to administrator or somebody else.

## 2.2 Changing user interface

Adaptation of user interfaces has been a problem considered for a long time [8]. Another goal is to provide an adaptable user interface at any moment offering next features:
1. Support for a number of different interfaces for each panel;
2. Allow users to switch between interfaces modes at any time, even in the middle of a command;
3. Switch between interfaces smoothly and naturally;
4. Make it easy for the user to learn how to use the different interfaces.

# 3 Adaptable user interface Solution

Based on the problem formulation we designed the next structure:

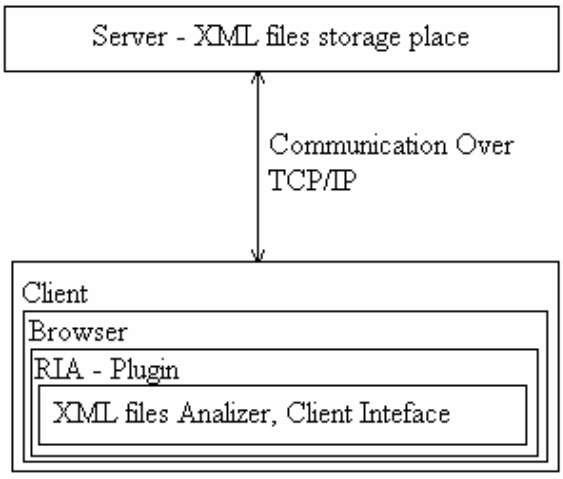


Fig. 1 – System design

XML files that define client adaptable interface are considered panels and all the panels are loaded in the main panel. Actually it's a graph structure where each node of the graph it is a panel as is showed in the next figure (Fig. 2).

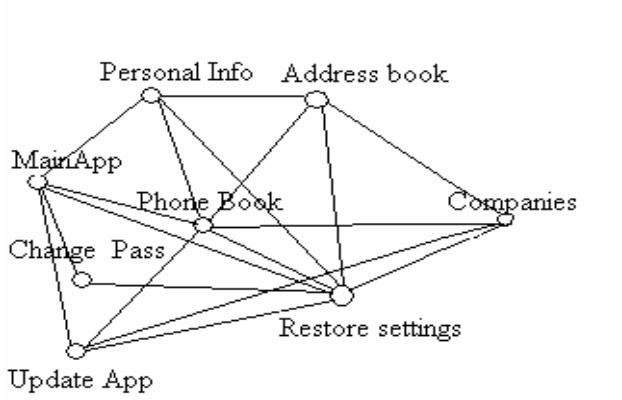


Fig. 2 – Panels interconnectivity

Each panel from this graph defines all the elements needed to generate the interface. Depending on complexity of element described it can be defined as an attribute or as a node. For example (Example 1) name of panel it is defined in attribute, but mouse event on click is defined as a separate node.

```
<xml>
 <components>
  <PopUpWindow backgroundColor='#397D02'
borderAlpha='0.81' height='114' id='NewPass'
idPanel='9054' isPopUp='true' layout='absolute'
panelType='ChangeAccountPass' title='Change User
Pass' ver='1.0.2' width='207' x='191' y='502'>
    <Macheta dataProviderSource='xmlNode:Macheta'
id='Macheta'></Macheta>
    <Form height='100%' width='100%' y='2'>
     <FormItem label='Old Pass' required='true'>
      <TextInput displayAsPassword='true' id='OldPass'
width='130' x='73'></TextInput>
     </FormItem>
         ……................
      <Button eventsSource='xmlNode:Send' height='22'
id='Send' label='Send' tabChildren='false' width='54'
x='95'></Button>
    </Form>
  </PopUpWindow>
 </components>
 <events>
  <Send><click> <switch menu='parent'>
<function>Application.socket.sendMessage(Macheta.getP
ass)</function>
     </switch> </click>  </Send>
 </events>
 <dataProvider>
  <Macheta>
         ..............
  </Macheta>
 </dataProvider>
</xml>
```

Example 1 –Panel definition

From this example we

## 3.1 Customization

Using XML files to generate the user interface we provide the following type of adaptability:

1. Global customization: Users can make global changes to a series of objects or to a single one.
2. Particular customization: Users should be able to make changes to one object without affecting other objects that have the same type.

When a global customization is required all objects that have the same type in all the XML files are modified and when a particular customization is require only one object it is affected.

There is a lot of reason for heaving a personalizable interface. Each user, even for users that have similar skills in using computers want to use a custom interface rather than a static predetermined one.

Personal or group needs can be done without the work of information technology department.

All the users prefer to have the control of the application instead of requesting all the modifications needed to administrator or somebody else.

One of the most important problems in design adaptable user interface is to provide a lot of options to customize and in the same time to offer an easy way to manage these options. Solution adopted by us is to provide a friendly interface to each property, style, event that can be customized. For example colors are presented as a panel and the user only have to choose one, numeric values are also represented in a box where the value can be changed directly but also with a scroll and selecting a point on the scroll line the corresponded value will be selected.

In example 1 we have defined button with the label “Send”, applying global customization to buttons mean that all tags named “Button” well be affected by user modification, also, we can make a particular customization to button “Send”.

## 3.2 Updating interface

Because XML files are used to store useful data interface we only have to replace existing previous XML file.

Studies made on 175 users that are using different interfaces that can provide custom updating show us that only less than 10 percents are using these facilities, but 75 percents consider that it is very important to have this feature.

After the study the conclusion was that these user adaptable interfaces have to include updating facilities and in default mode should do all the operations needed automatic. Also, capabilities of reverse some operations or reset to default facility it is very important.
Based on this studies and conclusions XML files that provide updating facilities has to include parameters to define what are the objects, properties, styles and events that have to be updated in any situation and what should be based on user options.
As was showed in example 1, in panel definition we have attribute "ver" where is defined the current version of panel. Using this attribute we can automatically check if there is a new version of the panel and based on update specifications a part of the attributes will be used as it is, or replace by attributes from update panel. The same this happening with objects that can disappear or new objects can be introduced in panel.

### 3.3 Using more interfaces

The idea of having more than one interface, with one that is personalized easily by the user, and putting the user in control of switching between interfaces was proposed by McGrenere and Moore [6]. They make a study based on people that used Microsoft Office and discovered that peoples need more that one interface to an application.
As is described in [7] an interface can be broken down into four levels:
*Concepts and Tasks* are taken from the underlying legacy Web model.
The *Abstract Interface* provides a UI representation common to all the RIA platforms without any kind of spatial arrangement, look&feel or behavior, so all the devices that can run RIAs have the same Abstract Interface.
Then the *Concrete Interface* may be optimized for a specific device. Concrete Interface is divided into three Presentation levels: Spatial, Temporal and Interaction Presentation.
Last one, the *Final Interface* provides the code generation of the modeled application.
Let' suppose that we have an application to manage address book with al the features needed. Even this interface it is very useful when we are using a laptop, can be very difficult to us to use the same interface in our phone browser. To solve this problem, we determining the devise type that is calling the interface and offer the specific XML files for this device.
To offer a specific set of files for user interface based on devise type we defining a property to each XML file to define interface type that should use this file. Also, based on a similar property, we can use different interfaces even on the same devise type.
The Final Interface automatically generate the user interface depending on the chosen technology: Flex, Ajax.

## 4 Conclusion

This paper introduces a modern concept design to generate customer interfaces adaptable, providing storage of all information necessary in XML files that are taken by the client and the authentication process generates client-side interface client.
To our knowledge this is the only framework based on rich internet application capable of generating a user adaptable interface in the moment of client login by analyzing XML files.

*References:*

[1] Anthony Jameson, The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications (2nd ed.), *Lawrence Erlbaum Associates*, 2008.
[2] M. Linaje, Juan C. Preciado and F. Sánchez-Figueroa, Title of the Paper, *Springer Berlin / Heidelberg*, Vol. 4607, 2007, pp. 226-241.
[3] Jin, J., Sang, N. and Liu, Y., XML-based user interface customization and dynamical modification of the embedded Linux system, *Journal of the University of Electronic Science and Technology of China,* Vol. 36, No. 3, 2007, pp. 510-513.
[4] Jin, J., Sang, N. and Liu, Y., User interface management with XML, *Journal of Computer-Aided Design and Computer Graphics*, Vol. 16, No. 4, 2004, pp. 566-571.
[5] D. Funke, J. Neal, and R. Paul, An approach to intelligent automated window management, *IJMMS*, Vol. 38(6), 1993, pp. 949–983.
[6] McGrenere, J. and Moore, G, Are we all in the same "bloat"?, *Graphics Interface 2000*, pp. 187-196.
[7] Limbourg Q., Vanderdonckt J., Michotte B., Bouillon L., Lopez V., UsiXML: a Language Supporting Multi-Path Development of User Interfaces, *9th IFIP Working Conference on Engineering for HCI*, Vol. 3425, 2005, pp. 207-228
[8] E. A. Edmonds, *Adaptive Man-Computer Interfaces, in Computing Skills and the User Interface*, Academic Press London, 1981