XML-Based Interface Model for Socially Adaptive Web-Based Systems User Interfaces

Janusz Sobecki

Department of Information Systems, Wrocław University of Technology Wyb.
Wyspińskiego 27, 50-370 Wrocław, POLAND, sobecki@pwr.wroc.pl

Abstract. The differences among the population of the web-based systems users have very significant consequences in their interaction preferences. Many web-based systems offer different personalization mechanisms that enable users to adjust systems' interfaces to their personal needs, however, quite many users don't want to bother themselves doing it or are not able to make reasonable settings. In the paper the XML-based interface model for socially adaptive user interfaces is presented. This model enables personalization representation and consensus based interface reconciliation among the population of personalized interfaces.

Keywords: xml-based interface model, adaptive user interface, web-based system

1 Introduction

Web-based systems are nowadays used for many different reasons: to inform, to teach, to promote or to agitate. They also concern with many areas of people's interests. The user population of those systems is usually very differentiated and that's why it's members could have different information needs and interaction preferences. Today it is obvious for almost everyone that web-based systems are more appreciated by their users if their interfaces are well designed. But building a single user interface that is uniquely effective for each of these systems and for all the users is rather impossible [16].

The problems of user interfaces are addressed by many different scientific areas. Traditional approach has been worked out by the area of the Human Computer Interaction (HCI) and it suggests designing an interface according to the future user profile as well as to the system platform characteristic [9,14]. Obviously such solutions could not be fully applied in the today's web-based systems because of the differences that concern users and platforms they are using. The interface personalization is a commonly used solution in these circumstances [8].

The personalization enables to set three basic elements of the system: the system information content, the way the information is presented and how the information is structured [6]. For example in the very popular web portal — Lycos users are able to add and remove boxes with information (content), move them round the screen and change their sizes and colors (presentation) and making it the starting page (structure).

During the personalization process users are usually obliged to enter some demographic data about themselves, such as name, address, occupation and interests. These types of data are called the user data and are used to model the user. Usually all types of data concerning users that are gathered by the system could be also called the user profile. The user profiles could then be stored in the system supplier's servers or on the client side in cookies files in form of some entries [15]. The user profile could also contain the so-called usage data that may be observed directly from the users interaction with web-based systems, such as URL's of visited pages, links followed by the user and data entered by the user into the forms on the pages [4].

The alternative solution to the system personalization mechanism is offered by the implementation of interface agents. They help to automate very different user tasks and in this way help him or her to interact with many different information systems [5]. In this paper the problem of users interaction with web-based systems is addressed using another approach i.e. the social adaptive hypermedia methods [6].

2 The Architecture of the Socially Adaptive Web-Based Systems User Interfaces

In this chapter the general architecture of the socially adaptive web-based systems users interfaces will be presented only for short, in more details it has been presented in the following works [12,15,19].

The web-based system users may fall into three categories. First group is comprised of the users who have already used the system on the particular platform. Second group is made up by the users who have used the system already but on the other computer platform and in third group there are the users who have never used this particular system before.

In the first case, it is not much to do for the adaptation mechanism, because every user of the system, while using it, gathers experiences and acquires habits, so introduction of any automatic changes should be restricted to only such cases when the user asks for them or the computed usability function values are extremely low.

The second case, because of its dual character, is very problematic. On the one hand, the web-based system is already known for the user and the user is aware of his or her own information needs, how they are fulfilled and the user does not welcome any changes. On the other hand, different platforms have usually different characteristic of input and output devices. For example, for a typical PC we have an input by means of a 101 (or so) keys keyboard and a mouse, and color 1024x768 display as an output. For the modern handhelds, however, we have a touch screen and stylus together with some special keys as an input and maximum high color 320x240 display as an output. When we take into consideration other platforms such as: information kiosks, set-top boxes and navigation systems mounted in cars, we can encounter other specific differences in input and output devices characteristics. So in the adaptive interface construction for a new platform we should preserve as much as possible from the former platform, for example the information content, its structure but also the way of its

presentation. In many cases, however, it is not possible, so it is suggested that at least these two former elements, included in the interface profile description, should be preserved. As for the latest element, i.e. the information presentation should be adaptively constructed as shown below.

In the third case, the interface should be automatically constructed for this particular user and this particular platform. It is assumed that web-based systems have a dynamic population of users located in a distributed environment. These web-based systems form a multi-agent system that communicates with their users. Agents collect both: the user data and the usage data of each particular system in form of the user profile that is transferred to the server. The users are able to personalize their systems. The quality of these settings is automatically measured by the appropriate interface usability measures [9,14]. The system personalization settings together with the usability measures values form the interface profile that is also sent to the server. Any novel user of the particular system is first classified according to the corresponding user profile [12], in this way the user is assigned to the particular class of users. Then the interface profiles belonging to the same class are used to create, by means of consensus methods [10,15], the interface profile for that particular user. The system quality measures and user classification will be described below. The problem of the personalization settings will be addressed in the following section.

The domain of the HCI [9,14] has worked out many methods for the interface evaluation but all of them are not straightforward and cannot be measured by a simple function. Quite often different empirical methods are used but also analytical methods or their mixture are applied. We must remember that in the socially adaptive web-based systems the usability factors values must be evaluated in an automatic manner. The usability is compound of several factors [9], such as: the speed of performance of the specified activity, incidence of errors, users' ability to correct errors, users' efforts to learn how to use the system, users' satisfaction with the system, etc. These factors, however, can be quite difficult to measure.

The user classification is a very well known problem that has been addressed for at least 40 years by the specialists from many fields, i.e. the information retrieval [3]. Nowadays, in the era of the e-economy and the Information Society, researches in this area are continued by many specialists from: HCI, user modeling and marketing [7].

The process of user classification results in grouping heterogeneous set of users $U = \{u_1, \dots, u_n\}$ into disjoint subsets of the set U by some measure of similarity or distance. Finding an appropriate measure, as well as a user representation, are key problems of users classification. The process of classification could be computationally difficult, even with exponential complexity with respect to the number of elements to be classified [2,3], so practically other sub-optimal solutions have to be used. They are usually based on the selection of some initial partition as for example in the Dattola algorithm [3]. In the architecture proposed here for the centroids could serve, for example the profiles of randomly selected users who personalized the system interface by their own. Then for each vector (user profile) the similarity function between the vector and each centroid is determined. The vector is joined to the class with the centroid that has the

highest value of the similarity function and also higher than assumed threshold, those below are assigned to the class of so-called isolated elements. Then for each class the centroids are recalculated and the process is repeated until any of vectors changes their class assignment. Finally all vectors from the class of isolated elements are assigned to the classes with the highest similarity function values.

3 XML-Based Interface Model

Because of the great differences among the users as well as the system platforms, the interface model to be used in the adaptive web-based systems should be very flexible in reflecting those differences in terms of some formalities that could be implemented in the chosen web technology. Firstly, the model should be easy to use and offer powerful tool for the system designer in the design of possibly wide range of interactive web-based systems [17]. Secondly, it should enable users of the web-based systems to personalize them, that mean to set them according to their information needs and interaction preferences. Finally, the model should enable determination of the representation of the set of interface profiles, that is the key element of the architecture proposed above.

In the following subsections the interface profile, the distance function between the interface profiles and the method for the representation determination in the set of the interface profiles will be presented.

3.1 Interface Profile

The interface profile is the element of the overall interface model, which task is to represent the actual hypermedia interface in the design, personalization and generation processes. The profile was first presented in the work [18]. In this paper a new modified version of the profile is presented.

To represent the interface we can use several different models known from the literature. We can use for example quite simple Pawlak [13] information system model, where the system S is a quadruple $S = \langle X, A, V, \rho \rangle$, X is a set of objects, A set of attributes describing these object, V set of attributes values and $\rho: X \times A \rightarrow V$ is an information function.

In many cases, however, it is more convenient to use multi-valued version of the Pawlak information system, where the information function is defined as follows: $\rho: X \times A \rightarrow 2^V \setminus \{\emptyset\}$. This is because of the nature of many attributes that describe an interface, i.e. topics of interest, sound types of different signals or other relevant information, which could be represent by set of values instead of an atom value. Moreover, in some cases the set of the attribute values could be ordered and repeated. For example the order of topics appearing on the screen could be quite important as well as the possibility to represent repeated values: template identifiers of some screen regions or soundtracks to be played in the background like in a jukebox. The case of ordered sets with repetitions seems to be the most general one, i.e. any other example: single value and subset of values could be treated as special cases of this one.

Nowadays object-oriented design is the most popular paradigm in the information system design. This is also the case in the user interface design. This type

of structure could be then easily represented in form of a XML document [1]. XML has become recently the standard for exchanging any information among different applications or even their layers and can be directly implemented in wide range of user interfaces implementations.

Different attributes that describe user interfaces could have hierarchical dependencies and could be represented in an object-oriented manner or by a tree-like structure. Then attributes and their values are attached to the tree nodes and edges reflect dependencies among attributes and their values. In consequence we can model the every instance of a user interface as the following structure $I = \langle N, E, A, V, n_0, \sigma, \theta, \delta \rangle$, where

- N — is a set of nodes,
- $E \subset N \times N$ — is a set of edges between nodes, where the first node is called a father and the second a child,
- A — is a set of attributes,
- V — is a set of attribute values and $\Pi'(V)$ is a set of all ordered subsets with repetitions of values from V ,
- n_0 — is the root node,
- $\sigma: N \rightarrow A \times \Pi'(V)$ — is a description function,
- $\theta: A \times \Pi'(V) \rightarrow 2^A$ — is the attribute mapping function,
- $\delta: A \times (V \cup \{\text{null}\}) \times (V \cup \{\text{null}\}) \rightarrow [0, 1]$ — is a distance function.

The structure to form a tree must have the following properties: there is only a single root node, there is only a single edge connecting two different nodes, all nodes except the root node have only one father each and all of them share the same predecessor i.e. the root node.

The description function σ assigns each node a pair attribute and ordered subset of their values. The attribute mapping function θ that assigns each pair: attribute and subset of their values the subset of the attributes that could be placed in the child nodes.

The distance function δ should be given by the system interface designer and fulfill all the distance function conditions but not especially all the metrics conditions [10]. The interface designer should specify the distance function between all pairs of atom values, together with the empty value, of all attributes. The distance function values could be enumerated or given in any procedural form. These values enable to find distances between ordered sets of values and also are necessary to determine the distance between any of two instances of the interface. The procedure for determining this distance can be found in [11], the values determined by this procedure falls in $[0, 1]$ interval and fulfils the distance function conditions.

3.2 Example of an Interface Profile

To represent the interface instance, as it was generated by the system or modified in the personalization process some elements for each web-based system should have been determined in advance, i.e. the set of attributes A , the set of their values V , the attribute mapping function θ , and distance function δ . These sets and functions are equal for every instance of the interface profile of that

particular system. What differs one instance from another is: the set of nodes N , the set of edges E , the start node n_0 , and the description function σ .

Let us consider a typical web-based information system, which goal is to promote, advertise and inform about a particular car model. It is obvious that this type of a system is intended for even hundreds of thousands of users from many different countries all over the world. These users may be of completely different age (teenagers, adults, elder people) and have different sex. So, it is essential that this type of system should be able to offer different interface profiles for different users, because of their different information needs and various interaction preferences.

It is possible to consider all the elements of the interface profile in the model presented above, i.e. the information content, its structure and presentation. However, in the following example only its presentation part is considered.

The sample system has the following parameters (see also Fig. 1):

- $A = \{\text{root, resolution, button_style, font_size, font_style, bckg_type, bckg_color, bckg_image, sound, sound_track, sound_effect, volume}\};$
- $V = V_{\text{root}} \cup V_{\text{resolution}} \cup V_{\text{sound}} \cup V_{\text{sound_effects}} \cup V_{\text{sound_track}} \cup V_{\text{volume}} \cup V_{\text{bckg_type}} \cup V_{\text{bckg_color}} \cup V_{\text{bckg_image}} \cup V_{\text{button_style}} \cup V_{\text{font_size}} \cup V_{\text{font_style}},$ where
 $V_{\text{root}} = \emptyset,$ $V_{\text{resolution}} = \{640 \times 480, 800 \times 600, 1024 \times 768, 1280 \times 1024\},$
 $V_{\text{sound}} = \{\text{yes, no}\},$ $V_{\text{sound_effects}} = \{\text{effect1, effect2}\},$
 $V_{\text{sound_track}} = \{\text{track1, track2}\},$ $V_{\text{volume}} = \{0, 1, 2, 3, 4, 5\},$
 $V_{\text{bckg_type}} = \{\text{color, image}\},$ $V_{\text{bckg_color}} = \{\text{white, blue, black}\},$
 $V_{\text{bckg_image}} = \{\text{stars, circles}\},$ $V_{\text{button_style}} = \{\text{text, graphic_static, animated}\},$
 $V_{\text{font_size}} = \{1, 2, 3, 4, 5\},$ $V_{\text{font_style}} = \{\text{arial, georgia, times}\};$
- $\theta(\text{root}, \emptyset) = \{\text{resolution, sound, bckg_type, button_style, font_size, font_style}\},$
 $\theta(\text{bckg_type, image}) = \{\text{bckg_image}\},$ $\theta(\text{bckg_type, color}) = \{\text{bckg_color}\},$
 $\theta(\text{sound, no}) = \emptyset,$ $\theta(\text{sound, yes}) = \{\text{sound_track, sound_effects, volume}\};$
 $\theta(\text{sound_track, } _) = \emptyset,$ $\theta(\text{sound_effects, } _) = \emptyset,$ $\theta(\text{volume, } _) = \emptyset,$
 $\theta(\text{button_style, } _) = \emptyset,$ $\theta(\text{font_style, } _) = \emptyset,$ $\theta(\text{font_size, } _) = \emptyset;$
- $\delta(\text{volume, } x, y) = |x - y| / |\max - \min|,$ where $\max$ and $\min$ are maximal and minimal values accordingly;
 $\delta(\text{volume, } x, \text{null}) = 1/2,$ $\delta(\text{sound_track, track1, track2}) = 0.7,$
 $\delta(\text{sound_track, track1, null}) = 1,$ $\delta(\text{sound_track, track2, null}) = 1.$

Because of the lack of a space the other values of the distance function will not be given. The information function values for this particular instance are given in the tree nodes presented in Fig. 1.

The tree structure shown in the Fig. 1 could be easily presented in form of XML document. This document is described by the interface profile structure formally given above in the section 3.1. It is also possible to use other formalisms to represent such structure, i.e. XML Schema [20]. XML Schema as DTD (Document Type Definition) is used to declare the XML document structure but is more powerful one. With XML Schema in contrary to the DTD it is possible to declare also types and patterns that could be use to declare attributes, their values (also in form of ordered subsets with repetitions of values) as well as information function σ , attribute mapping function θ and distance function values

Fig. 1. Instance of the interface profile in form of the tree structure

δ . Unfortunately the XML Schema of the interface profile structure is outside the scope of this paper, here there is only the XML document that represents the interface instance and it will be given below.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<root>
  <resolution value="800x600"/>
  <sound value="yes">
    <sound_track value="track1 track2"/>
    <sound_effects value=""/>
    <volume value="2">
  </sound>
  <button_style value="text"/>
  <bckg_type value="image">
    <bckg_image value="circle"/>
  </bckg_type>
  <font_size value="3"/>
  <font_color value="blue"/>
</root>

```

3.3 Distance Determination among Interface Instances

The key element in the adaptive web-based system user interfaces model is its ability to measure the distance among different instances of user interfaces. Those instances may be given by the users themselves in the process of the interface personalization or determined by the adaptive procedures. In the interface model

given above only the distance among attribute values are given, but the model assumes that attribute values may be in the form of ordered sets of values with repetitions, so it is necessary to determine the distance among them too. An example of this procedure could be found in work [18], this procedure is called **set_distance** in the program below. Having this it is possible to find the distance among the interfaces instances modeled by the tree structures t_1 and t_2 . This distance is measured by the recursive function **tree_distance** described below, where the lv denotes the level of the node in the tree structure (equals 1 at start-point) and $root(t)$ denotes the root node of the tree t . In the function construction it was assumed that when attributes in the roots of sub trees are different the distance between these sub trees should be equal, independently of all other values of this sub tree, the maximal value 1 (in the whole distance however, it is divided by the level of the root of the sub tree and the number of sub trees in this level). This assumption is a consequence of the fact that only comparable values could be compared. For example, there is no use to compare integer value of background color with the volume value. In the instance of the interface represented by a tree structure attributes and their values depend on the attributes and corresponding values that are placed higher in the tree so there is no use to compare them too.

```

function tree_distance(t1,t2,lv)
BEGIN
  (a1,w1)= $\sigma$ (root(t1))
  (a2,w2)= $\sigma$ (root(t2))
  IF a1<>a2 THEN RETURN 1
  ELSE IF  $\theta(a1,w1)<>\theta(a2,w2)$  THEN RETURN set_distance(w1,w2)
  ELSE
    BEGIN
      c:=1/| $\theta(a1,w1)$ | /* $\theta(a1,w1)=\theta(a2,w2)$  */
      k:=lv+1
      s:= $\sum$  set_distance(w1,w2)
      RETURN c*s+1/k* $\sum$  tree_distance(t'1, t'2,k) /*where t'1
      and t'2 are all sub-trees of the trees t1, t2 starting
      from edges derived from the root node */
    END
  END
END

```

The value of distance determined by the above function falls in $[0,1]$ interval and fulfils the distance function conditions.

3.4 Representation of the Set of Interface Instances

To solve the problem of determining the representation in the set of interface instances we should also have the values of usability function $u(j)$ assigned to each of the interface instance. Let assume that the usability function values falls also in the $[0,1]$ interval and the value 0 denotes completely unusable interface and 1 denotes ideally usable interface. Then to find the interface instance for a particular user by means of consensus method, we must find the interface instance i that's the sum of the distances to all other instances j from the

profile, i.e. those instances that are assigned to users belonging to the same (as our user) class C , multiplied with their utility function value $u(j)$ is minimal:

$$\min \left(\sum_{j \in C} u(j) * \text{tree_distance}(tr_i, tr_j, 1) \right). \quad (1)$$

This problem could be computationally difficult. We can have two approaches to solve it. First we can reduce the problem space only to those instances that belong to the set of all known instances (i.e. they were used already by other users from the same class of users) but the solution could be far from optimal one, or second, we can consider all possible interface structures that could be constructed out of the set of attributes A and values from the set V which are valid in the model for a particular system. However, in the works [10,11] it was proven that for some distance functions we could reduce the search space only to those attribute values that are already presented in the profile and this suffices to find satisfactory representation of the profile.

4 Conclusion

Adaptive user interfaces are becoming more and more popular among different web-based information services [6]. Unfortunately most of the solutions are commercial ones and quite often their methodologies are rather confidential. Considering the social perspective in the adaptive systems enables to follow those interface settings that are not only often used but also those that appeared to be the most effective, owing to application of usability measures.

In this paper the XML-model for adaptive interfaces has been presented. Using this model several web-based systems interfaces are being implemented already in Macromedia Flash. Macromedia Flash is very popular tool for multimedia web-based system implementation and its recent version MX is quite well suited for XML documents processing even without direct implementation of such mechanisms as XML Schema, DTD or XSL which the application developer has to implement in Flash or Action Script.

References

1. Bray T., Paoli J., Sperberg-McQueen C. M., Maler E.: Extensible Markup Language (XML) 1.0 (Second Edition), W3C. 6 October (2000) <http://www.w3.org/TR/REC-xml>.
2. Brown S. M. et al.: Using explicit requirements and metrics for interface agent user model for interface agent user model correction. In: Proceedings of the second international conference on Autonomous Agents, Minneapolis, Minesota, USA (1998) 1-7.
3. Dattola R. T.: A fast algorithm for automatic classification. Report ISR-14 to the National Science Foundation, Section V, Cornell University, Department of Computer Science (1968).
4. Fleming M., Cohen R.: User modelling in the design of interactive interface agents. In: Proc of the Seventh International Conference on User Modeling: (1999) 67-76.

5. Hedberg S. R.: Is AI going mainstream at last? A look inside Microsoft Research. *IEEE Intelligent Systems* **3/4** (1998) 21–25.
6. Kobsa A., Koenemann J., Pohl W.: Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships. *The Knowledge Engineering Review* **16** (2), (2001) 111–155.
7. Maglio P. P., Barrett R., Campbell C.S., Selker T.: SUITOR: an attentive information system. In: *Proc. of the 2000 Int. Conf. on Intelligent User Interfaces* (2000) 169–176.
8. Mobasher B., Cooley R., Srivastava J.: Automatic personalization based on Web usage mining. *Comm of the ACM*, **43** (8), (2000) 142–151.
9. Newman W. M., Lamming M. G.: *Interactive system design*. Addison-Wesley, Harlow (1996).
10. Nguyen N. T.: Conflict Profiles' Susceptibility to Consensus in Consensus Systems. *Bulletin of International Rough Sets Society* **5**(1/2): (2001) 217–224.
11. Nguyen N. T.: Consensus system for solving conflicts in distributed systems. *Information Sciences* **147** (2002) 91–122.
12. Nguyen N. T., Sobecki J.: Consensus problem of user classification in interactive systems. In: P. Grzegorzewski (et al., eds.): *Soft Methods in Probability, Statistics and Data Analysis. Series Advances in Soft Computing*. Physica Verlag Heidelberg (2002) 346–355.
13. Pawlak Z.: *Information Systems — Theoretical Foundations*. PWN Warsaw (1983).
14. Pearce J. et al.: *Human-computer interaction*. Addison-Wesley, Harlow (1996).
15. Sobecki J., Nguyen N. T.: Consensus-based adaptive user interface for universal access systems. In: Stephanidis C. (ed.) *Proceedings of 9th Int. Conf. on Human-Computer Interaction and 1st Int. Conf. on Universal Access in Human-Computer Interaction*. LEA London vol 3 (2001) 112–116.
16. Sobecki J.: One suits all — is it possible to build a single interface appropriate for all users? In: Grzech A., Wilimowska Z. (eds.) *Proceedings of the 23rd Int. Scientific School ISAT*, PWR Press Wroclaw (2001) 125–131.
17. Sobecki J.: Interactive multimedia information system planning. In: Valiharju T. (ed.), *Digital Media in Networks. Perspectives to Digital World*. University of Tampere (1999) 38–44.
18. Sobecki J.: Interface model in adaptive web-based system. In: Cellary W., Iyengar A. (eds.): *Internet Technologies, Applications and Societal Impact*. Kluwer Academic Publishers, Boston (2002) 93–104.
19. Sobecki J.: Adaptive interface for hypermedia applications. In: *Proceedings of VIPromCom-2002 — 4th EURASIP — IEEE Region 8 International Symposium on Video Processing and Multimedia Communications*, 16–19 June 2002, in Zadar, Croatia: (2002) 147–152.
20. van der Vlist E.: *Using XML Schema*. O'Reilly & Assoc. Inc. (2000)
<http://www.zml.com/pub/a/2000/11/29/schemas/part1.html>.