Interface Model in Adaptive Web-based System

Janusz Sobecki
Department of Information Systems, Wroclaw University of Technology, Wroclaw, Poland
sobecki@pwr.wroc.pl

Abstract:

In the paper the model of a social adaptive web-based system user interface is presented. In the model the object-oriented hierarchical representation is used to reflect the properties of the interface. The model was constructed to be suitable for: a user interface design, an interface personalization, as well as determination of a representation of a set of interfaces.

Key words: interface model, adaptive web-based systems, interface representation

1. INTRODUCTION

Web-based systems are nowadays one of the most widely used type of information systems. They found their application in almost all areas of human activities: commerce, industry, education, banking or entertainment. Effectiveness, availability and well-designed user interfaces of today's web-based systems are becoming essential for every successful application [11,15]. There are however quite many reasons why it is very difficult to accomplish these requirements. Three of them, which are believed to be the most important, could be distinguished. These are the following reasons: great differences among user population of the web-based systems [12], great number of the available web-based systems and differences between system platforms that are used to access these systems.

In the design of every interactive system, the user model should be determined prior to any further step [7,10,14]. However the ever-increasing number of the web-based systems users also brings the increasing

---

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35616-7_23
W. Cellary et al. (eds.), *Internet Technologies, Applications and Societal Impact*

differences among them. These users are living in all parts of the world; are members of different cultures; have different nationalities; belong to different generations; etc. Obviously, all these anthropological, as well as cognitive differences that could be encountered among these users, have influence on their information needs and interaction habits. In the consequence it is very difficult to model them properly [7,13].

2. INTERFACE PERSONALIZATION AND INTERFACE AGENTS

Interface personalization is a common feature of many web-based systems, i.e. portals and many other information or entertainment systems. The personalization could be made in many different ways [6]. Usually in the beginning of this process users are asked to enter some demographic data, such as name, age, address, occupation, interests, contacts, etc. Then the users are able to customize the interface to their needs and capabilities. Generally, the users could set three basic elements of the system: the system information content, the way the information is presented to them and how it is structured.

Users, when setting the interface view can choose for example: the number of columns for the text display, one of the several graphical and style templates, the color template as well as choose the music sample played in the background. All these interface settings form the interface profile that could be stored in the system files located on the server side or in the cookies located on the client side.

System personalization is the utility that is provided for the user from the system side. However, there are also solutions that are more dependent from the user, and are design to serve primary him or her. They are called the interface agents. P. Maes defines an interface agent as an agent that acts as a kind of intelligent assistant to a user with respect to some computer application [16]. The MIT agents act primarily by observing their users, and by application some machine learning mechanisms. For each new situation, the agent computes the distances, between the current state and all appropriate states stored in the memory. Together with these past states corresponding actions taken by the user in the past are stored. Then, for this new situation, the recalled action is selected which state resembles the most, or in other words has the smallest distance between, this new situation [4].

Interface agents in order to predict properly the user intents must work on accurate user model, but quite often they can be inaccurate. There could be many different reasons for this, but most important are: differences between users and users' behavior changes over time [1].

The interface agents could be very useful in the situations, where there are standard repeatedly completing user actions. But when users enters the system for the first time they can't offer him very much. The same is with personalization utility that requires a lot of efforts from the user to adjust the system to his or her particular needs. Quite often users are not eager to spend too much time on system settings, especially when they use the system only occasionally, or even once in a quite long period of time. Usually users are not fully aware of the system functionality and interface settings. Finally, they could have problems with choosing the optimal settings for them, especially when there are many different possible settings.

It is possible, however, to find among the more experienced users of any particular system, those who are somehow alike to the user using this system for the first time, and then follow their settings for him or her, or in other words use social adaptive methods for interface construction.

3. INTERFACE ADAPTATION

Adaptive user interfaces are defined as software that improves its ability to interact with a user by constructing a user model based on partial experience with that user [6]. In the social, or in other words collaborative, adaptive interface architecture the user model is constructed not only basing on his individual experience but also taking into consideration the experiences of some other users that resembles somehow this particular user.

Hypermedia applications have usually a dynamic population of users located in distributed environment. In the social adaptive interface architecture the user's platforms form a multi-agent system that communicates with them. Agents collect information about the environment, i.e. users, in the form of the user profiles. From the other hand the users may personalize different applications and these settings are stored in the interface profile.

Social adaptive interface architecture is designed to construct automatically the interface of each hypermedia application for each novel user of that application that is better than the standard one. When the user wants to start the application for the first time, the agent sends his user profile to the application server. On the server side its profile is classified and then all the interface profiles of the users whose user profiles belong to the same class are used to construct the interface profile for that particular user. Then this automatically prepared interface profile is used to create the real interface of that application that is presented to the user.

The user profiles may consist of different users "experiences" with using the Web, for example: URL's of visited pages or user queries send to the

Web systems. These pages are usually in form of HTML document's, with quite deal of natural language words [3]. These words placed in the vector could serve as a user profile then using one of the method presented in [2], i.e. Datola method, it is possible to classify users.

In the Datola method it is assumed that class centers are known, for example they could be arbitrary selected. Then for each vector (user profile) the similarity function (quite many of them are mentioned in [2]) between the vector and each of the class centers is determined. The vector is joined to the class with the center that has the highest value of the similarity function, but over the specified threshold. Unclassified vectors are left isolated. Then for each class new centroides are determined and the process of determining the similarity function values are repeated for all the vectors. The process is stopped when no changes in object assignment is encountered.

The effectiveness of the user's interaction with the system could be measured by means of the utility function [7]. Hypermedia applications could be built and maintained for different reasons, mainly they are made for making money but there are also many ways for doing this. This could be the web portal that earns money on advertisements, Internet shop that sells different goods or Internet auction servers that enables exchange of different goods between their users, and many other ways. Each of these applications has different goals and so different facts express their achievement, so the utility function should be specified particularly for every individual application.

Having the class of similar users, interface profiles with the utility function values assigned to them it is possible to construct the interface profile for the novel user, who belongs to the same class. The interface profile consists of different parameters that are used in the particular interface construction. For every system this process should be designed separately, so here only some general ideas concerning this matter can be presented. The potential interface attributes that could be selected raises from quite simple as: fonts, their styles, colors and sizes, background colors or graphics, buttons in form of icons or texts; or more complex as: parameters of used set of fonts (their correlation), used whole interface templates (as for example themes in Microsoft FrontPage) or style of buttons animation. These could be also more general parameters such as used metaphors, or attributes describing information content and its hierarchy.

Automatic adaptive interface construction for classified users could be made in several different ways. In works [11,13] adaptive interface construction for general access web-based systems with application of the consensus methods [8] have been presented. But also other methods could be considered, for example Bayesian belief networks [5]. The easiest solution to this problem is the selection for each user one of the already existing

interface profile of the user from the same class. This could be the profile with the highest utility function or the interface profile of the user whose user profile resembles the most the class centroid [2]. We can also consider such selection that takes into account these two criteria in the same time. The method is quite simple but does not take into account experiences of the whole users community (at least from this particular class) as it is done in consensus-based method [11]. Finally the new interface profile is sent back to the agent, where the concrete interface is generated.

4. INTERFACE MODEL

The interface model to be used in the adaptive hypermedia should have several properties. First it should be easy to use by the system designer that means it should be easy to represent the actual interface in the design and implementation process in the form of so called the interface profile. We must also remember that in the adaptive system the interface is generated on-line and the model should enable it. Secondly we assume that the system interface should be also personalized, so the user settings should be easily represented in the model and modified on-line. And finally, the third, the model should enable finding the representation of the set of interfaces or in other words to find consensus among the interfaces.

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

4.1 Interface profile

The interface profile is the element of the overall interface model, it's role is to represent the actual hypermedia interface in the design, personalization and generation processes.

The object-oriented models are currently the most popular among system designers. Many programming languages such as C+, Java, JavaScript or new C# are built on the object paradigm, as well as many script languages used in multimedia authoring tools such as Lingo from Macromedia Director or ActionScript from Macromedia Flash. Also recent standard in the web-based information exchange XML enables hierarchical, object-based representation.

To represent the interface we can use several different models known from the literature. First we can use simple Pawlak information system model [9]. In the model the information system S is a quadruple:

$<X,A,V,\rho>$, where

- X is a set of objects,
- A is a set of attributes describing these object,
- V is a set of attributes values,
- $\rho: X \times A \rightarrow V$ is an information function.

In many cases, however, it is more convenient to use the multi-valued version of the information system [9], where the information function differs from the former one and could be given as follows: $\rho: X \times A \rightarrow 2^V - \varnothing$. The other version was introduced to model better many system attributes that have multi-valued nature, that describe i.e. topics of interest or keywords describing documents.

In the interface modelling we can assume that in some cases the set of the attribute values could be not only multi-valued but also ordered and repeated. For example, the order of topics appeared on the screen could be quite important as well as the possibility to represent repeated values i.e. template identifiers of some screen regions or soundtracks to be played in the background like in a jukebox.

The attributes that describe the interface could have hierarchical dependencies and as such could be represented in object-oriented manner or in tree-like structure, where attributes are attached to the tree nodes and values to the tree edges. It could be also assumed that the edges could have the empty value and finally all the leaves have a null attribute.

In consequence we can model each system interface as the following structure:

$<N,E,A,V,n_0,\sigma,\theta,\delta>$, where

- N – is a set of nodes,
- $E \subset N \times (\Pi'(V) \cup \{null\}) \times N$ – is a set of edges 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 sets with repetitions of values from V,
- n_0 – is the root node,
- $\sigma: N \rightarrow A \cup \{null\}$ – is a description function,
- $\theta: A \times (V \cup \{null\}) \rightarrow 2^A \cup \{null\} - \varnothing$ - is the attribute mapping function,
- $\delta: A \times (V \cup \{null\}) \times (V \cup \{null\}) \rightarrow [0,1]$ – 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 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 distance function δ should be given by the system interface designer and fulfil all the distance function conditions but not especially the all the

metrics conditions [8]. The values of the distance function should be determined for each attribute and its every pair of atom values plus empty value. The distance function values could be enumerated or given in any procedural form. In the tree representing the interface to the edges are assigned ordered subsets with repetitions instead of atom values, so we must also be able to determine the distance among them. To do so me must introduce the notion of the element rank r, i.e. the position of element in the ordered set. The distance $d(w_1,w_2)$ where $w_1,w_2 \in \Pi'(V) \cup \{null\}$ and there are such edges $e_1=(n_1,w_1,n_2)$ and $e_2=(n'_1,w_2,n'_2)$, where $\sigma(n_1)= \sigma(n_1)=a$, is determined by the following function

```
function set_distance(w1,w2)
BEGIN
d:=0
k:=1/(|w1|+|w2|)
s:= min(|w1|,|w2|)
FOR i:=1 TO  s
  BEGIN
  find      v1∈w1      and      v2∈w2      such      that
  q:=k*δ(a,v1,v2)+k*|r(v1)-r(v2)| is minimal
  d:=d+q
  remove v1 and v2 from w1 and w2 respectively
  END
s:= max(|w1|,|w2|)-s
FOR i:=1 TO s
  BEGIN
  d:=d+k*δ(a,v,null)
  remove v from w1 or w2
  END
RETURN d
END
```

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

4.1.1 Example of an interface profile

To represent the interface implementation, 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 interface

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

Let as consider a typical web-based information system, which goal is to promote, advertise and inform about particular make of car (i.e. Mercedes, Volvo, Peugeot or Toyota just to name some well known ones). It is obvious that this type of system could be used even by millions of users from many different countries all over the world, being of completely different age (from young boys, through teenagers, adults to elder people) and having different sex. So, we believe that this type of system should be able to offer different interface profiles for different users, because of their different information needs and different interaction preferences. The sample system has the following parameters (see also fig. 1):

- the set of attributes A={root, window_size, sound,template, sound_track, sound_effect,volume}
- the set of attribute values $V=V_{root} \cup V_{window\_size} \cup V_{sound} \cup V_{template} \cup V_{sound\_track} \cup V_{sound\_effect} \cup V_{volume}$, where
  - V_{root}={null}
  - V_{window_size}={640x480,800x600,1024x768,1280x10024}
  - V_{sound}={null}
  - $V_{template}$={blocks,bars,highway,kids,safari}
  - V_{sound_track}={track1,track2}
  - V_{sound_effect}={effect1,effect2}
  - V_{volume}={0,1,2,3,4,5};
- attribute mapping function values, where the underscore sign "_" stands for all possible values
  - θ(root,_)={window_size,sound,template},
  - θ(template,_)={null},
  - θ(volume,_)={null},
  - θ(sound,_)={sound_track,sound_effects},
  - θ(sound_track,null)={null} and for the rest of values θ(sound_track,_)={volume},
  - θ(sound_effects,null)={null} and for the rest of values θ(sound_effects,_)={volume}
- the distance function values for all different attribute values
  - δ(window_size,x,y)= |a-b|/m , where a is the rank of the value x and b of the value y in the ordered set of m possible resolutions ordered increasingly;
  - δ(template,x,y)= |a-b|/m, where a is the rank of the value x and b of the value y in the ordered set of m possible templates ordered as shown above;

- δ(volume,x,y)=|x-y|/|max-min|, for x and y different from null, where max is the maximal and min is minimal values; when one of the values equals null then δ(volume,x,null)=1/2;
- δ(sound_track,track1,track2)=0.7; δ(sound_track,track1,null)=1; δ(sound_track,track2,null)=1;

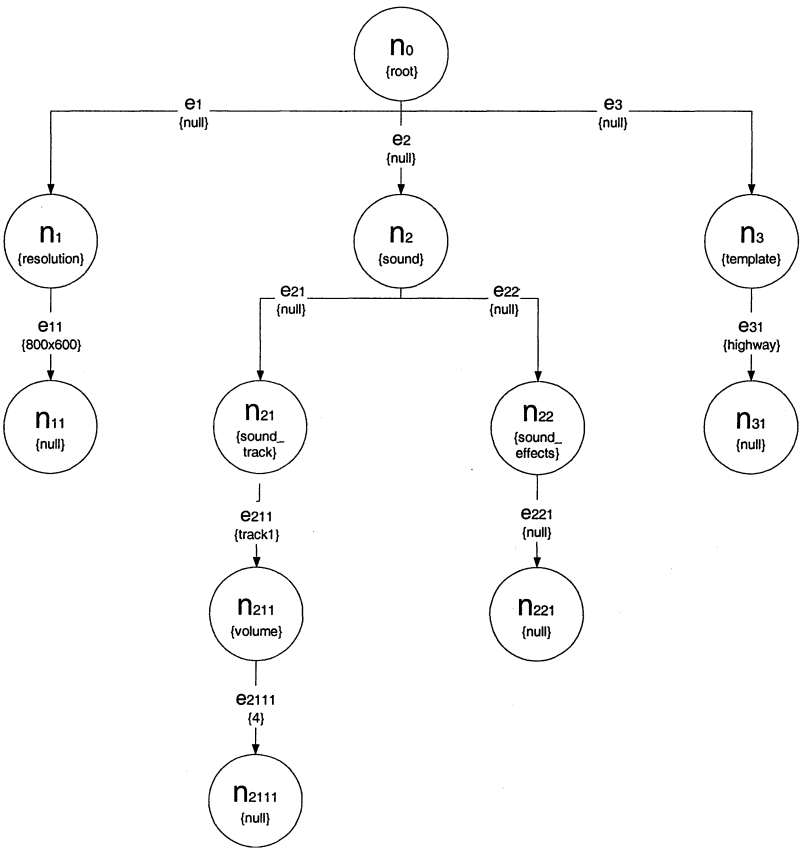

Figure 1. Interface profile tree structure

4.2 Distance determination among interfaces

The distance among the system interfaces modelled by the tree structures t_1 and t_2 is measured by the recursive function **tree_distance** described below.

Let lv denotes the level of the node in the tree structure (equals 0 at start-point) and root(t) denotes the root node of the tree t.

```
function tree_distance(t1, t2,lv)
BEGIN
a1=σ(root(t1)) and a2=σ(root(t2))
IF a1<>a2 THEN RETURN 1
ELSE IF a1=a2=null THEN RETURN 0
    ELSE
    BEGIN
    c:=1/|θ(a1)| /*θ(a1)=θ(a2) */
    k:=1/(lv+1)
    s:=Σ set_distance(w1,w2) /*sum of distances
    between sets assigned to all edges starting
    from root(t1) and root(t2) respectively*/
    RETURN c*s+k*Σ tree_distance(t'1, t'2,lv+1)
    /*where t'1 and t'2 are all sub-trees of the
    trees t1, t2 starting from such nodes which edges
    have equal values assigned to them i.e. w1=w2*/
    END
END
```

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

4.3 Representation of the set of interfaces

To solve the problem of finding the representation in the set of interfaces we must have the values of utility function *u(j)* assigned to each of the interface. Let assume that the utility function values falls in the [0,1] interval and the value 0 denotes completely useless interfaces and 1 denotes ideally useful interface. Then to find the consensus we must find the interface *i* that's sum of the distances to all other interfaces j belonging to the same class *C* multiplied with their utility function value *u(j)* is minimal:

$$\min(\Sigma_{j \in C} u(j)*\text{tree\_distance}(tr_i, tr_j,0))$$

This problem could be computational difficult. We can have two approaches to solve it. First, we can reduce the problem space to only those interfaces that belong to the set of interfaces, 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.

5. CONCLUSIONS

The problem of clustering people and assigning their classes corresponding properties is very old. In 60-ties Jonathan Robin found correlation between peoples living place denoted by the zip code determines peoples purchasing behaviours but also other behaviours such as political or leisure time preferences. These researches was strongly adapted by e-commerce, internet users are often asked to give their personal data including zip code while registering in many web-based system.

In this paper the method for social adaptive interface construction was presented. The problems of user model construction and user classification as well as interface utility function determinations were shortly addressed. The interface structure, the distance determination and finding the representation were described in more detailed manner. The presented method has not been verified yet but other findings and also empirical research in other areas shows promising perspectives for such solutions also in the human computer interaction area.

6. REFERENCES

1. Brown SM et. al. (1998) Using explicit requirements and metrics for interface agent user model for interface agent user model correction. In: Proc. of the second international conference on Autonomous Agents, Minneapolis, Minesota, United States pp. 1-7.
2. Dabrowski M, Laus-Maczynska K (1978) Information retrieval and classification. Survey of methods (in polish). WNT Warszawa.
3. Estivil-castro V, Yang J (2001) Categorizing visitors dynamically by fast and robust clustering of access logs. Lecture Notes on Artificial Intelligence 2198: 498-507.
4. Fleming M, Cohen R (1999) User modelling in the design of interactive interface agents. In: Proceedings of the Seventh International Conference on User Modelling: 67-76.
5. Hedberg SR (1998) Is AI going mainstream at last? A look inside Microsoft Research. IEEE Intelligent Systems 3/4: 21-25.
6. Kobsa A., Generic User Modeling Systems. User Modeling and User-Adapted Interaction 11: 49-63, 2001.
7. Newman WM, Lamming MG (1996) Interactive system design. Addison-Wesley Harlow.
8. Nguyen NT (2001) Conflict profiles' susceptibility to consensus in consensus systems. Bulletin of International Rough Sets Society 5(1/2): 217-224.
9. Pawlak Z (1981), Information Systems – Theoretical Foundations. PWN Warszawa.
10. Peerce J et. al. (1996) Human-computer interaction. Addison-Wesley Harlow.
11. Sobecki J, Nguyen NT (2001) Consensus-based adaptive user interface for universal access systems. In: Stephanidis,C (ed.) Proc. of 9th Int. Conf. on HCI and 1st Int. Conf. on Universal Access in HCI. LEA London vol.3 pp. 112-116.
12. Sobecki J (2001) One suits all - is it possible to build a single interface appropriate for all users? In: Grzech A, Wilimowska Z (eds) Proc. of the 23rd Int. Scientific School ISAT, PWr Press Wroclaw pp. 125-131.

13. Sobecki J, Nguyen NT (2001) Adaptive user interfaces for general access web systems. In: Bubnicki Z, Grzech A (eds) Proc. of the 14th International Conference on Systems Science., PWr Press Wroclaw vol. 3 pp. 70-77.
14. Sobecki J (1999) Interactive multimedia information planning. In: Valiharju T. (ed.) Digital Media in Networks 1999, Univ. of Tampere pp. 38-44.
15. Spolsky J (2001) User interface design for programmers. Apress LP.
16. Wooldridge M, Jennings NR (1995) Intelligent agents: theory & practice. Knowledge Engineering Review 100(2): 115-152.