A NEW ALGORITHM FOR SELF-ADAPTING WEB INTERFACES

Bogdan Vintila, Dragos Palaghita and Maria Dascalu
Economic Informatics Department, University of Economics Bucharest, Calea Dorobanti Street, Bucharest, Romania

Keywords: Self-adapting web interface, Software development, Interface design, Dynamic web interfaces, Statistical analyses.

Abstract: The paper proposes a new improved algorithm for creating hierarchies of features and options for self-adapting web interfaces against the common one used by many applications. The user interface concept is presented. Types of user interfaces are described. Quality characteristics of the user interfaces are analyzed. Ways of fulfilling these quality characteristics while keeping the costs low are discussed. Advantages and disadvantages of self-adapting and static web interfaces are given. The most common algorithm for creating hierarchies of features and options is described and analyzed. The advantages and disadvantages of the proposed algorithm are discussed. New directions for the development of the self-adapting web interfaces are highlighted.

1 INTRODUCTION

User interfaces are pieces of software that ensure the interaction between the program's logic and the user. In order to be successfully utilized by users, a software product needs not only to have flawless functionality but also to expose its features in an accessible way. Even if the program has rich features that work perfectly, a deficient user interface leads to the fail on market. This is explained by the users' lack of time for learning how to use new software and thus they orient towards easy to use software. User interfaces experienced a great evolution from the beginnings of the computing age. As user interfaces are used for the communication between the user and the application they must facilitate the selection of processing options, data input and output. The user interfaces started with the *batch interface* in 1945 (Wikipedia, 2009). In 1969 the batch interface was replaced by the *command-line user interface*. The command-line user interface has the advantage of giving the user access to all commands and parameters. The main disadvantage is that the user must know the syntax of all options and commands he wants to access. This led to the apparition of the Graphical User Interface (GUI) in 1981. Nowadays the command-line user interface addresses highly trained professionals that need precise control and access without the performance overhead of the graphical interface.

There are many types of user interfaces (Wikipedia, 2009):

- Graphical User Interfaces;
- Web User Interfaces; are used by web applications; the user utilizes a Internet browser to access the application; data input is made through a form generated by the server; results' visualization is made also through web pages generated by the server and loaded by the users' Internet browser (Hall Mary W., 2008); nowadays these are widely used as web applications have many advantages against standalone ones;
- Command-line Interfaces;
- Touch User Interfaces;
- Gesture Interface;
- Multi-screen Interfaces;
- Motion Tracking Interfaces;
- Voice User Interfaces.

Standalone applications are difficult to update. The process depends on the user, so compatibility to earlier versions must be ensured. Web applications, on the other side, are easy updateable as the application resides on one or more physical servers. Also, web applications can be accessed regardless of the location or machine. Presently, web applications

Vintila B., Palaghita D. and Dascalu M.
A NEW ALGORITHM FOR SELF-ADAPTING WEB INTERFACES.
DOI: 10.5220/0002773400570062
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

tend to replace standalone applications in more and more domains. Given the importance of these applications, their interfaces are of high importance as they ensure the communication between the user and the application's logic.

In (Esko Juuso, 1997) the role of neural networks, genetic algorithms and linguistic equations in the development of user interfaces are described. Many challenges in the development of adaptive interfaces and also many research directions are given in (Langley, 1999).

Adapting the interface to the needs of the user reduces its complexity and improves efficiency. Awareness is analyzed as an evaluation metric in order to obtain the reduced visual complexity of interfaces (Findlater & McGrenere, 2009).

The study presented in (Letsu-Dake & Ntuen, 2009) shows clearly that the adaptive interfaces are better in terms of time to achieve the goal under fault conditions, fault detection time, number of system failures and fault detection rate.

In (Pietschmann et al., 2009) a study of the business processes is made through the eye of user interfaces. User interface services provide many rich, reusable components for building user interfaces.

The importance of the adaptive interfaces for special domains such as the automotive industry is highlighted in (Amditis et al., 2006). A methodological framework for optimizing the human machine interfaces is presented and the results of its implementation are discussed.

The concept of *Universal Access* is presented by (Stephanidis, 2001) and also ways of building systems that ensure interfaces capable of adapting according to the context. Universal access is attained through the adaptive interfaces.

The importance of the adaptive interfaces is analyzed in (Savidis & Stephanidis, 2004) for a wide range of devices. The unified user interfaces described by the paper are capable to self-adapt at run-time according to the needs and requirements of the current user and device. The development process of unified user interfaces is also described by the authors.

A methodological approach for modeling adaptation decisions and for solving the problem of integrating existing as well as acquired knowledge in the decision module of an adaptive interface is proposed in (Zarikas, 2007). The model uses influence diagrams and it provides a method of encoding user and context information as well as other factors that are involved in the decision making process. The paper also presents an illustrative example of the analyzed modeling method.

The notions of user profile, interface profile and the compound usability are discussed in (Nguyen & Sobecki, 2003). Their role in the development of the adaptive interfaces is also analyzed. Using consensus-based methods, the authors build interface profiles appropriate to classes of users.

2 QUALITY CHARACTERISTICS OF USER INTERFACES

All interfaces must fulfill quality requirements in order to give the users the maximum degree of satisfaction. Some of the most important quality characteristics are (Ivan et al., 2008):

- clarity;
- conciseness;
- familiarity;
- responsiveness;
- consistency;
- attractiveness;
- efficiency;
- forgiveness.

These characteristics, if fulfilled at the same time, ensure the success of the user interface. Anyway, by increasing one of the characteristics it is possible to lower others so achieving a balance between then is a complex and long process (Julio Abascal, 2008). Considering the application's target group it is possible to satisfy some requirements with lower levels of some characteristics leaving additional time to increase the level of more important ones.

The measurement of the quality characteristics is usually a hard task as there are no clear procedures of measurement for each of them. Let us consider responsiveness. For this characteristic the obtained value upon measurement depends on the user if no measurement procedure is stated. If the application is unresponsive for one second after the user gives a command, the user might consider it lacks a good response time or might even not notice the delay. If a measurement procedure is defined there is no room for subjective judgment. If the procedure states an interface is responsive if it has delays less than 1 second, the considered case lacks responsiveness. If the procedure states a time of more than 2 seconds, the considered case is responsive.

Table 1: Advantages and disadvantages of self-adapting and static web interfaces.

	Self-adapting web interfaces	Level of the user	Static web interfaces	Level of the user
Advantages	Clear Light Efficient Attractive Simple Easy to use Change depending on context Change in time Low bandwidth usage Suggest additional features	End-user End-user End-user End-user End-user End-user End-user End-user End-user End-user	Efficient Detailed Doesn't change in time or according to context Allow workflow automation Detailed control	Professional Professional Professional Professional Professional
Disadvantages	Don't allow detailed control Force users to request additional features	Professional Professional and End-user	Overcrowded High complexity Can't be efficiently used Unattractive High bandwidth usage	End-user End-user End-user End-user End-user

3 STATIC VS. SELF-ADAPTING WEB INTERFACES

The pass from static user interfaces to the self-adapting ones was done in order to facilitate the use of software by untrained users and also increase the efficiency. Self-adapting interfaces also decrease the operating time by displaying the right features in the right spot and hiding the unnecessary ones (Ion Ivan, 2009). Self-adapting user interfaces are based on studies regarding the users' comportment in a certain application or when trying to solve a certain problem. These have a predictable comportment as the frequency indicators don't change during application's use or only in a small proportion. There are also interfaces that study the user's comportment and are dynamically built based on the indicators calculated using these data. E-Commerce sites suggest products based on user's comportment or similarities with other users (Sharifi & all, 2004). Menu options hide until the first use. Options are showed in the order they are used most.

Static user interfaces are used for applications that have few features and options or for ones that address highly trained professionals. Static interfaces are difficult to use for end-users, but professionals prefer them as they can automate workflow. The self-adapting web interfaces have the advantage of conserving precious bandwidth. By determining the most used features of the application, the graphical elements corresponding to the other features don't usually have to be loaded at the start. If necessary, the user can request the load of additional components as he uses the application. The lower the used bandwidth is, the higher the user's experience gets.

Advantages and disadvantages of the self-adapting and static web interfaces are given in (Table 1).

As seen in (Table 1) advantages and disadvantages are taken into account considering the interface's target group.

Detailed control is a must be for the highly trained professional, but for the end-user is not important. Also the interface's ability to change in time is an advantage for the end-user but a disadvantage for the professional that quickly learns the features' options and location. As users differ in training level, the same interface has positive or negative impact in use. The higher the training level, the more interfaces the user can use and overcome their disadvantages. Both types of interfaces have advantages and disadvantages so the users must choose the one that makes him more efficient.

4 COMMON ALGORITHM

Self-adaptive web interfaces must adapt their behavior from an application's run to another and even during the work session (Savidis & Stephanidis, 2004). By recording and analyzing user's comportment it is possible to obtain such results. There are many algorithms to predict user's comportment based on past actions (Gena C., 2007). The most common assumes the following:

Let *F* be a set of *n* features the user can access in a given application *A*. An application's feature is defined by a succession of steps that, through the processing of the input data, lead to wanted results.

For example, the print process that is present in many software products, is a feature of them.

$$F = \{f_1, f_2, \ldots, f_n\} \quad (1)$$

Let Q be a set of n frequencies associated to the set of features. Feature one, f_1 is associated to q_1, f_2 to q_2 and so on.

$$Q = \{q_1, q_2, \ldots, q_n\} \quad (2)$$

On the base on Q set a hierarchy of the features is built. The most accessed features rank the highest (Ivan et al., 2009).

Let us consider the A application was never used and all frequencies are 0. Considering this, all features have the same priority. At the first use, feature f_1 is used two times, f_3 is used once and f_{n-1} is used once.

At the second use, the Q set has the following values:

$$Q = \{2,0,1,0, \ldots, 0,1,0\} \quad (3)$$

These lead to the following hierarchy:

$$H = \{f_1, f_3, f_{n-1}, f_2, f_4, f_5, \ldots, f_{n-2}, f_n\} \quad (4)$$

Considering H, at the second run f_1 is the first one in the selection list, f_3 is the second, f_{n-1} is the third and the ones with 0 frequency follow. This way, the user is more likely to have the feature he wants to access at the top of the selection list. The efficiency of the algorithm increases with every use of the application.

The possibility of having individual interfaces for each user, rather than the whole application, is given by storing data for each user and building the interface based on that data (Brusilovsky, 2001).

5 THE IMPROVED ALGORITHM

A new algorithm is proposed to improve user experience. The algorithm is based on calculating scores and ranking features on the bases of dependencies between features.

Let $D(f_i, f_j)$ be the function that returns the dependency value between features i and j. The highest the value of $D(f_i, f_j)$ is, the higher is the probability that the user selects feature j after selecting feature i. A dependency table is obtained by calculating the D function for all features. The main diagonal elements are equal to 0 as the selection of a feature can't determine the repeated selection leading to an infinite loop. When a feature is accessed by the user, the score of the features that depend by this one are updated accordingly to the dependency's strength. This leads to better scores for features that were not selected, but have a very high probability of being selected.

Let $D(f_1, f_2) = 0.6$, $D(f_2, f_4) = 0.2$ and all the other dependencies be 0.

Taking into consideration this system of ranking, the above dependencies and the number of feature access above, the resulting Q set is

$$Q_1 = \{2,1.2,1,0.24,0, \ldots, 0,1,0\} \quad (5)$$

leading to the following H_1 hierarchy

$$H_1 = \{f_1, f_2, f_3, f_{n-1}, f_4, f_5, \ldots, f_{n-2}, f_n\} \quad (6)$$

Comparing H to H_1 we see that the order of features is different, f_2 being the second feature in the list before f_3 and f_{n-1}. The interface updates according to this hierarchy either at set moments, either when the changes that arise between the old arrangement and the new one grows above a set value. Using the dependencies system better forecasts of the user's comportment can be made, but only if the dependencies are determined using very large datasets for analyses. The use of incorrect dependencies between features results in incorrect selection lists. These cause poor user experience as instead of quickly solve its problem, the user tries to find the desired feature. As the user utilizes more and more the application, the frequency set Q gets to be more and more significant for the user's comportment and the application will make better forecasts regarding future user's actions.

A very important issue is choosing the D function that returns the dependencies between values. A very simple, yet effective, way of establishing dependencies is by analyzing users' behavior in existing applications. Let us consider an e-commerce application is to be built. For this we can analyze the data on users' behavior that other e-commerce applications already have. After filtering and clustering the data, dependencies of the clusters are determined and these are used for the dependencies set of features (Lau & Horvitz, 1999). Even if the e-commerce is borderless, the percentage of local users is greater than the percentage of global users. This can lead to good predictions on new users' behavior only for local ones. This can be avoided by studying data from many applications worldwide.

Compared to the basic algorithm, this one improves the success rate of having a desired feature closer to the user.

6 ALGORITHM VALIDATION

A sample application is being currently developed to exemplify the self-adapting web interface using the dependency enabled rank computing algorithm.

As the algorithm states, the ranking system is based on the addition to the base frequency of the dependency score. The dependencies used are not bidirectional thus $D(f_i, f_j) \neq D(f_j, f_i)$. This is normal because action a determines action b, but action b can't determine action a. The application is to implement real functionality and is to be released to the public to test the algorithm's validity. The time needed by users to select options is to be recorded and the obtained data is to be analyzed leading to the algorithm's validity or invalidity. Short times between the selections of features indicate a good prediction of the user's comportment. Long times for feature selection indicate the algorithm's failure in providing qualitative forecasts. Collected data must be preprocessed as to include in the analysis only data obtained by users after several use of the application when they understand the application's adaptive comportment.

7 CONCLUSIONS

With time many types of interfaces were developed. The diversity is given by advantages of each type of interface for a certain type of application or process. Self-adapting web interfaces are of great future as more and more users have low training. These interfaces allow the untrained users to use the informational systems at a basic and advanced level without training. The common algorithm for creating hierarchies can be improved by taking into account dependencies between the collectivity's elements. The self-adaptive interfaces are of great importance in domains such as: e-learning, e-governance, office suites, operating systems, mobile applications.

Future research includes color coding the features so that the color best perceived by the human eye is associated with the feature most probable the user will access, the second color in the perception hierarchy is associated to the second most probable feature and so on. Future research also aims at the use of cameras to keep the user's eyes under observation and detect the screen zones the user focuses most and thus placing there the most accessed features and options. By detecting the direction of human gaze, the navigation within the interface is possible. The user focuses the desired option and based on his position, distance from the camera and previous configuring, the application detects the selected area and activates the feature.

ACKNOWLEDGEMENTS

This article is a result of the project „Doctoral Program and PhD Students in the education research and innovation triangle". This project is co funded by European Social Fund through The Sectorial Operational Programme for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies.

REFERENCES

Amditis, A., Polychronopoulos, A. & Andreone, L., 2006. Communication and interaction strategies in automotive adaptive interfaces. *Cognition, Technology and Work*, 8(3), pp.193-99.

Brusilovsky, P., 2001. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, pp.87-110.

Esko Juuso, K.L., 1997. Adaptive Interfaces and Soft Computing. *Studies in Informatics and Control*.

Findlater, L. & McGrenere, J., 2009. Beyond performance: Feature awareness in personalized interfaces. *International Journal of Human-Computer Studies*, 68(3), pp.121-37.

Gena C., W.S., 2007. Usability Engineering for the Adaptive Web. In *The Adaptive Web*. Springer Berlin / Heidelberg. pp.720-62.

HALL Mary W., G.Y.L.R., 2008. Self-Configuring Applications for Heterogeneous Systems: Program Composition Using Cognitive Techniques. *Proceedings of the IEEE*, 96(5), pp.849-62.

Ion IVAN, B.V.C.C.M.D., 2009. The Modern Development Cycle of Citizen Oriented Applications. *Studies in Informatics and Control*, 18(3).

Ivan, I., Vintila, B. & Palaghita, D., 2008. Quality metrics of citizen oriented informatics applciations. In *Forth International Conference on Applied Statistics*. Bucharest, Romania, 2008.

Ivan, I. et al., 2009. Collectivity's elements ranking. In *16th International Economic COnference IECS 2009*. Sibiu, Romania, 2009.

Julio Abascal, I.F.d.C.A.L.J.M.C., 2008. Adaptive Interfaces for Supportive Ambient Intelligent Environments. In *ICCHP 2008*., 2008. Springer Berlin / Heidelberg.

Langley, P., 1999. User modeling in adaptive interfaces. In *Proceedings of the seventh international conference on User modeling*. Banff, Canada, 1999. Springer-Verlag New York, Inc. Secaucus, NJ, USA.

Lau, T. & Horvitz, E., 1999. Patterns of search: analyzing and modeling Web query refinement. In *Proceedings of the seventh international conference on User modeling*. Banff, Canada, 1999. Springer-Verlag New York, Inc. Secaucus, NJ, USA.

Letsu-Dake, E. & Ntuen, C.A., 2009. A case study of experimental evaluation of adaptive interfaces. *International Journal of Industrial Ergonomics*, 40(1), pp.34-40.

Nguyen, N.T. & Sobecki, J., 2003. Using consensus methods to construct adaptive interfaces in multimodal web-based systems. *Universal Access in the Information Society*, 2(4), pp.342-58.

Pietschmann, S., Voigt, M. & Meibner, K., 2009. Adaptive rich user interfaces for human interaction in business processes. In *Lecture Notes in Computer Science*., 2009.

Savidis, A. & Stephanidis, C., 2004. Unified user interface development: the software engineering of universally accessible interactions. *Universal Access in the Information Society*, 3(3-4), pp.165-93.

Sharifi, G. & all, 2004. Location-Aware Adaptive Interfaces for Information Access with Handheld Computers. In *Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer Berlin / Heidelberg. pp.328-31.

Stephanidis, C., 2001. Adaptive Techniques for Universal Access. *User Modeling and User-Adapted Interaction*, 11(1-2), pp.159-79.

Wikipedia, 2009. *User Interface*. [Online] Available at: http://en.wikipedia.org/wiki/User_interface [Accessed 16 October 2009].

Zarikas, V., 2007. Modeling decisions under uncertainty in adaptive user interfaces. *Universal Access in the Information Society*, 6(1), pp.87-101.