A Development Toolkit for Unified Web-Based User Interfaces

C. Doulgeraki[1], N. Partarakis[1], A. Mourouzis[1], and C. Stephanidis[1,2]

[1] Foundation for Research and Technology – Hellas (FORTH)
Institute of Computer Science
Heraklion, GR-70013, Greece
{cdoulger,partarak,mourouzi,cs}@ics.forth.gr
[2] University of Crete, Department of Computer Science

Abstract. EAGER is a prototype development toolkit that allows embedding accessibility and ease of use for all potential users into Web-based artefacts. Web-based user interfaces developed by means of the EAGER toolkit incarnate the concept of *Unified User Interfaces* and exhibit adaptation behaviour with respect to diverse user abilities, requirements and preferences. Ultimately, the process of employing EAGER is significantly less demanding in terms of time, experience and skills required from the developer, than the typical process of developing for the "average" user.

Keywords: Unified User Interfaces, Adaptation, Design for All, User profiling.

1 Introduction

The Web has evolved into a continuously growing source of knowledge, information and services, potentially accessed anytime and from anywhere. Web users are potentially all citizens. Therefore *universal access* to the Web emerges as a fundamental requirement in the context of the Information Society [1], necessitating approaches that ensure accessibility and usability of Web-based applications for users with diverse characteristics and requirements. Still, the vast majority of Web applications today are designed for the "typical" or "average" user, while accessibility is - at best - addressed through conformance to guidelines, thus producing "one-size-fits-all" results that may not be optimal for some target use group, or necessitate additional assistive technologies.

Recent approaches to *universal access* and *design for all* have emphasised the central role of *user interface adaptation* towards satisfying the needs and requirements of diverse target user groups. So far, adaptation has been explored mainly in the context of independent applications for desktop environments. In the Web environment, adaptation techniques have mostly been applied at the level of user agents (e.g., the AVANTI browser [3]). However, such approaches are limited by the fact that the user, in order to gain access to the Web, must have the actual product installed on the computer used. On the other hand, intermediary agents, acting as filtering and transformation tools, have also been proposed for building alternative versions of Web content based on usability heuristics and accessibility recommendations. The practical

K. Miesenberger et al. (Eds.): ICCHP 2008, LNCS 5105, pp. 346–353, 2008.

exploitation of this concept (e.g., [4]) has highlighted a number of issues that tend to reduce the universality of the approach, leading to the development of specialised filters for each website parsed by an intermediary agent. Clearly, the diversity characterising Web users and usage, for instance in terms of technological skills and experience, interaction abilities and preferences, access platforms and input/output devices, application domains and user tasks cannot be addressed adequately by just "fixing" the rendered html output.

2 The Unified Web-Based User Interfaces Methodology

This paper presents the first approach worldwide targeted to support and provide the means for the development of inclusive *Web-based user interfaces* (WUIs) capable to adapt to multiple and significantly different user profiles and contexts of use. To this purpose, the *Unified Web Interfaces* (UWIs) method is proposed, building on the *Unified User Interface* (U^2I) methodology [2] for supporting the development lifecycle of user interfaces (UIs) capable of adaptation behaviour in terms of content, navigation, layout and user interaction models.

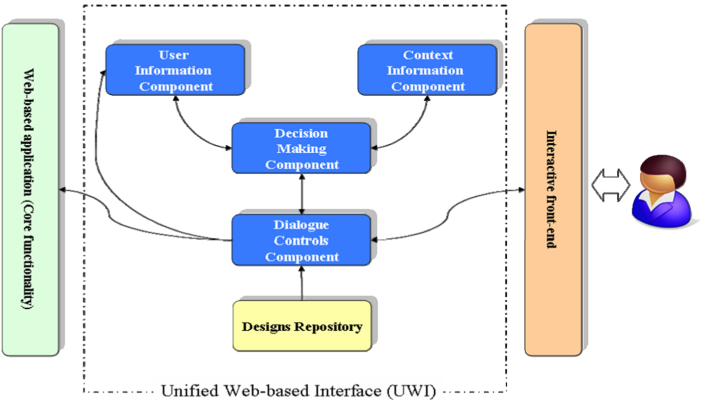

Fig. 1. Architecture of *Unified Web-based Interfaces* (UWIs)

The basic components of the UWI architecture (see Fig. 1) are:

- **The User Information Component (UIC)**; responsible for collecting and propagating attributes related to the characteristics of each specific user.
- **The Context Information Component (CIC)**; responsible for collecting and propagating attributes related to the context of use.
- **The Decision Making Component (DMC)**; in charge of the overall decision making regarding the conditional activation – deactivation of interaction modalities and interface elements.
- **The Designs Repository (DRE)**; a repository of alternative designs.
- **The Dialogue Controls Component (DCC)**; responsible for assembling the various elements into a concrete interactive front-end.

2.1 User Information Component (UIC)

The UIC (see Fig.2) acts as a server for collecting and providing information about user profiles. Each user profile contains attribute values automatically identified or specified by the user, both prior and during interaction. To collect such information during the interaction, a specific monitoring mechanism is used inside the UIC. The *User Profiles Repository* is a database of all users and the corresponding profile data records. On the other hand, the *User Components Repository* stores information regarding the conditional activation - deactivation of interactive elements per user as propagated by the DMC. To achieve bidirectional transmission of data, specialised Web Services and Logic are incorporated acting as a proxy class to the implementation underlying these two repositories. The *Interaction Monitoring Module* provides the mechanisms mentioned above for monitoring the interaction history of each user and inform accordingly the User Profiles Repository for future use. The data recorded by this module includes records of successful or unsuccessful completion of actions, the subjective preferences of navigation options, etc. The core element of the UIC is the Profiling Module, which is responsible for propagating User Profile information to the DMC, and additionally acts as an interface to the rest of the UWI components as well as to specialised profiling UI modules, such as:

- *The User Profiles Statistics UI module*, which presents statistics regarding the popularity of the various designs and settings available.
- *User Profile Selection UI module*, which enables the user to choose among predefined user profiles or to configure manually a new one.
- *User Profiles Administration UI module*, which allows site administrators to define predefined profiles and facilitate their main target user groups.

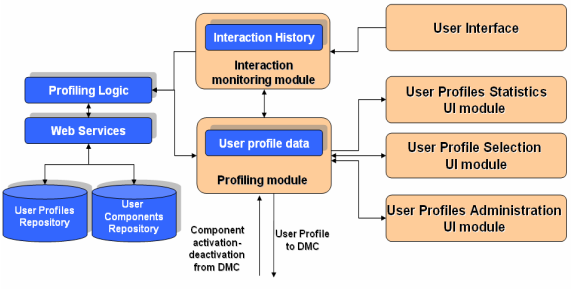

Fig. 2. The User Information Component (UIC) architectural model

Fig.3 depicts an example of the attribute value based user profile model. Similar considerations hold for the CIC presented in the next section.

2.2 Context Information Component (CIC)

The CIC is intended to collect and propagate context attribute values (machine and environment) of two types: (a) (potentially) static, meaning unlikely to change during interaction, e.g., browser and peripheral equipment; and (b) variant, dynamically

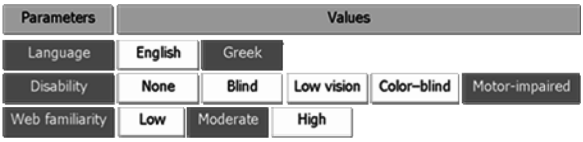

Fig. 3. Example of a User Profile instance

changing during interaction (e.g., due to environment noise, or the failure of particular equipment, etc.). A *Context Monitoring Module* that has the responsibility to monitor context changes and propagate this information to the *User Profiling Module* mentioned above. This module in turn enriches a *User Context Profile Repository* with these context specific attributes to be used in the process of decision making. Clearly, the attributes to be supported dynamically by CIC are quite limited, due to the current lack of methods for collecting such information from the client side.

2.3 Decision Making Component (DMC)

The DMC decides, in essence, when, why and how adaptation will occur. In other words, it entails the logic regarding the conditional activation and deactivation of alternative UI components according to user and usage attributes propagated by the UIC and the CIC. The core of this component consists of a number of implemented rules representing the design space of the user interface by mapping hierarchically various user attributes to appropriate alternative designs. For example, the decision logic for presenting links can be the following:

- if *web knowledge* belongs to {*high*}, then use *coloured links*;
- if *web knowledge* belongs to {*moderate*}, then use *underlined links*;
- else use *push buttons*.

2.4 Dialogue Controls Component (DCC)

The role of DCC is to apply the interface adaptations decided by the DMC and structure the final front-end of the underlying application using the selected dialog components. More specifically, this component (i) provides the implementation of the alternative dialog components of a self-adapting interface in the form of dynamic libraries; (ii) moderates and administrates the alternative dialog components; and (iii) maintains a record of user interaction with alternative dialog components.

2.5 Designs Repository (DRE)

The DRE component is populated with designs of alternative dialogues controls in a form of abstract design and polymorphism. Polymorphic decomposition leads from an abstract design pattern to a concrete artefact. U^2I design emphasises on capturing abstract structures and patterns inherent in the interface design, enabling incremental specialisation towards the lower physical-level of interaction, and making therefore possible to introduce design alternatives as close as possible to physical design [6]. This makes it easier to introduce at any stage additional values of design parameters

(e.g., considering new types users and contexts) without affecting the whole of the design space. Fig.4 depicts an example of polymorphic task hierarchy, illustrating how two alternative dialogue styles for an "upload file" task may be designed. Alternative decomposition "styles" are depicted in the upper part, and an exemplary polymorphic decomposition at the lower part. Fig.5 includes physical design annotation corresponding to the alternative styles.

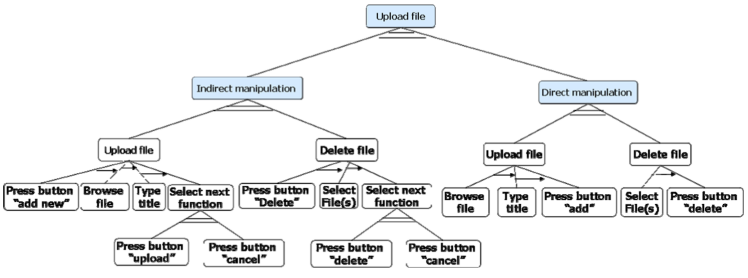

Fig. 4. The polymorphic task hierarchy concept

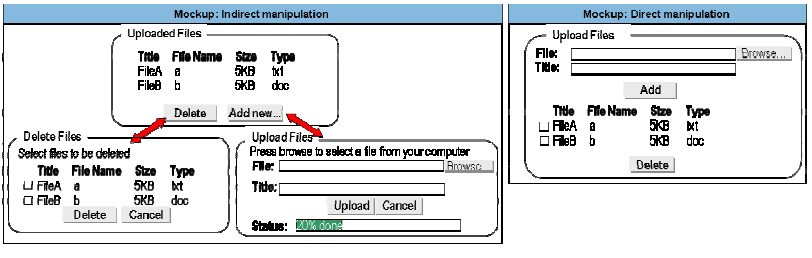

Fig. 5. Physical design alternatives (for File uploader)

In the depicted process, the following primary decisions need to be taken: (a) at which points of a task hierarchy polymorphism should be applied, based on the considered (combinations of) user- and usage-context- attributes; and (b) how different styles behave at run-time. This is performed by assigning to pair(s) of style (groups) design relationships. These decisions need to be documented in a design rationale that directly associates user- / usage-context- parameter values with the designed artefacts. As a minimum requirement, such a rationale should document [2]: related task, design targets leading to the introduction of the style, supported execution context, style properties and design relationships with competing styles.

3 The EAGER Toolkit

In order to facilitate Web developers in applying the proposed UWI method in practice, a prototype development toolkit, named EAGER[1], was developed. EAGER is an

[1] EAGER stands for "toolkit for embedding accessibility, graceful transformation and ease of use in Web–based products".

advanced library of the core UWI architecture components: User Information, Context Information, Decision Making, Dialogue Controls (activation/deactivation), of the primitive UI elements with enriched attributes (e.g., buttons, links, radios, etc.), of the structural page elements (e.g., page templates, headers, footers, containers, etc.), and of the fundamental abstract interaction dialogues in multiple alternative styles (e.g., navigation, file uploaders, paging styles, text entry). The EAGER toolkit has been developed in Microsoft® Visual C# .NET and according to the UWI framework briefly presented above. The technologies that were used for the development of the EAGER toolkit include:

- Microsoft Visual C# .NET for the implementation of the UI modules.
- Microsoft Visual C# .NET and XML for Business Logic and Web Services.
- Microsoft SQL server 2000 for the database implementation.

For the development of EAGER, a number of UI elements were designed and implemented in various forms (polymorphic task hierarchies) according to specific user and context parameters values. This phase provided input to the actual development process of EAGER, which involved the implementation of the alternative interaction elements and of the mechanisms for facilitating the dynamic activation - deactivation of interaction elements and modalities based on individual user interaction and accessibility preferences. The EAGER toolkit was then employed to develop an advanced portal[2] for the European Design for All and e-Accessibility (EDeAN) network as a proof of concept (see Fig.5). This effort provided valuable feedback on a number of issues, and proved the viability and usefulness of using EAGER in the development of large scale applications. A number of alternative evaluation techniques were applied to EAGER and to the developed prototype portal, including conformance to the W3C accessibility guidelines for Web content.

In summary, EAGER allows Microsoft® .NET developers to create interfaces that conform to W3C accessibility guidelines and which are able to adapt and interchange modalities, metaphors and user interface elements as appropriate for each individual user, according to profile information and context specific parameters. The process of employing EAGER is significantly less demanding in terms of time, experience and skills required from the developer than the typical process of developing Web interfaces for the "average" user, due to the flexibility provided for designing and implementing interfaces at an abstract task-oriented level. Using EAGER, designers are not required to be aware of the low level details introduced in representing interaction elements, but only of the high level structural representation of a task and its appropriate decomposition into sub tasks, each of which represents a basic UI and system function. In conclusion, the EAGER toolkit offers, not only the aforementioned benefits, but also opens a more promising direction. It is clear that using a standard UI toolkit, a monolithic interface is created, whereas by using the EAGER toolkit, dynamically adaptable interfaces are generated.

[2] http://hci-web7.ics.forth.gr/edean

Fig. 6. Default view of the homepage of the EDeAN portal

Another key feature of the EAGER toolkit is its ability to be extended and include an unlimited number of alternative interaction modalities and elements. This process mainly entails the design and coding of the alternative interactions styles. Then, they can be easily incorporated in the existing toolkit, simply by modifying the decision logic for supporting their conditional activation and deactivation. Additionally, existing Web applications or parts of applications implemented with .NET can be easily altered to encapsulate the EAGER toolkit attributes and, thereby, rendered accessible and usable for various user categories, including novice users, users of Assistive Technologies or portable devices, etc.

The proposed approach allows embedding in Web-based applications decision making logic and automatic adaptation facilities for the benefit of accessibility and better user experience. On the other hand, it has also proved that the proposed approach can produce Web applications that allow their users to choose themselves (i.e., customise) the designs they prefer most. Therefore, it is feasible to develop Web-based interfaces that can support and import alternative designs for fully accessible and personalised ways of interactions, without payoffs in terms of aesthetics or inclusiveness.

For detailed information regarding the proposed UWI method, the EAGER toolkit and the example portal of EDeAN the reader may refer to [7].

4 Conclusion and Future Work

A potential direction for future work concerns the integration of EAGER with a design support tool for the U^2I Design method (e.g., [5]). Such integration will result into a graphical tool to support the development process of UWIs from design to implementation, and allow extending easily and semi-automatically the EAGER user and context profiles and adaptation logic. Finally, another potential direction of future work is to render the EAGER toolkit a web service, so that Web developers using technologies other than .NET will be able to incorporate the EAGER adaptation logic into their artefacts, by providing an interface for defining profiles and receiving decisions regarding the activation – deactivation of alternative UI elements. Then, the developer would only have to implement, if not available, the proposed alternative designs in their own development environments.

Overall, the work presented here is considered as a significant contribution towards embedding accessibility, graceful transformation and ease of use for all in future and existing Web-based applications, and, ultimately, towards supporting individuals to fully participate in the knowledge society, especially people at risk of exclusion.

References

1. Stephanidis, C. (ed.)., Salvendy, G., Akoumianakis, D., Bevan, N., Brewer, J., Emiliani, P.L., Galetsas, A., Haataja, S., Iakovidis, I., Jacko, J., Jenkins, P., Karshmer, A., Korn, P., Marcus, A., Murphy, H., Stary, C., Vanderheiden, G., Weber, G., Ziegler, J.: Toward an Information Society for All: An International R&D Agenda. International Journal of Human-Computer Interaction 10(2), 107–134 (1998)
2. Savidis, A., Stephanidis, C.: Unified User Interface Design: Designing Universally Accessible Interaction. International Journal of Interacting with Computers 16(2), 243–270 (2004)
3. Stephanidis, C., Paramythis, A., Sfyrakis, M., Savidis, A.: A Case Study in Unified User Interface Development: The AVANTI Web Browser. In: Stephanidis, C. (ed.) User Interfaces for All, pp. 525–568. Lawrence Erlbaum, NJ (2001a)
4. Alexandraki, C., Paramythis, A., Maou, N., Stephanidis, C.: Web accessibility through adaptation. In: Proceedings of the 9th International Conference on Computers Helping People with Special Needs (ICCHP 2004), Paris, France, July 7-9, pp. 302–309. Springer, Heidelberg (2004)
5. Antona, M., Savidis, A., Stephanidis, C.: A Process–Oriented Interactive Design Environment for Automatic User Interface Adaptation. International Journal of Human Computer Interaction 20(2), 79–111 (2006)
6. Savidis, A., Stephanidis, C., Emiliani, P.L.: Abstract Task Definition and Incremental Polymorphic Physical Instantiation: The Unified Interface Design Method. In: Salvendy, G., Smith, M.J., Koubek, R.J. (eds.) Design of Computing Systems: Cognitive Considerations [Proceedings of the 7th International Conference on Human-Computer Interaction (HCI International 1997)], San Francisco, USA, August 24-29, vol. 1, pp. 465–468. Elsevier, Elsevier Science, Amsterdam (1997)
7. Doulgeraki, C., Partarakis, N., Mourouzis, A., Antona, M., Stephanidis, C.: Towards Unified Web-based User Interfaces. FORTH-ICS Technical Report, TR-394 (2007), `http://www.ics.forth.gr/publications/technical-reports.jsp?raey=2007`