User-Interface Modelling for Blind Users

Fernando Alonso, José L. Fuertes, Ángel L. González, and Loïc Martínez

School of Computing, Technical University of Madrid, Campus de Montegancedo, 28660-Boadilla del Monte, Madrid, Spain
{falonso,jfuertes,agonzalez,loic}@fi.upm.es

Abstract. The design of a user interface usable by blind people sets specific usability requirements that are unnecessary for sighted users. These requirements focus on task adequacy, dimensional trade-off, behaviour equivalence, semantic loss avoidance and device-independency. Consequently, the development of human-computer interfaces (HCI) that are based on task, domain, dialog, presentation, platform and user models has to be modified to take into account these requirements. This paper presents a user interface model for blind people, which incorporates these usability requirements into the above HCI models. A framework implementing the model has been developed and implemented in an electronic speaking bilingual software environment for blind or visually impaired people and in an educational system for children with special educational needs.

Keywords: human-computer interfaces, blind user interface model, accessible user interfaces.

1 Introduction

Apart from basic human-computer dialogue and design for all principles, adapting a graphical user interface for blind people involves some specific usability requirements: (1) the task has to be adequate given the capabilities of blind users (task adequacy), (2) the user interface has to provide a balance between the 2D access of sighted people and the 1D access of blind people (dimensional trade-off), (3) the user interface has to provide specific access for blind people to all the relevant user interface objects (behaviour equivalence), (4) the user interface has to avoid losing relevant semantic information (semantic loss avoidance) and (5) the interface has to deal with a wide variation in the functionality and programming of the assistive technologies for blind people (device-independency).

These requirements have an impact on all the models used in human-computer interface (HCI) development: the task, domain, dialog, presentation, platform and user models. Based on these requirements and the experience acquired by our team over the years, we have developed user interface model extensions that are presented in this paper. This paper is structured as follows. Section 2 briefly summarizes the related work on user interfacing for the blind. Section 3 describes our proposal for a user interface model for blind people. Section 4 presents the framework for blind user interface development based on the previous model. Finally, section 5 presents some concluding remarks.

2 Related Work

There are some user interface management systems (UIMS), toolkits and frameworks that can be used to design dual user interfaces (that is, graphical user interfaces that can also be used by blind people because they combine visual and non-visual modalities) [1]. This same philosophy can be applied to develop applications based on AJAX or other technologies, establishing the mechanisms for implementation [2]. To get an efficient final product, however, some formal interface design and implementation method has to be applied with these tools.

In any case, it is essential, as pointed out by Moreley [3], to examine the fundamental accessibility issues for blind people at length and define appropriate usability guidelines in order to design an interface suited for use by blind people. These guidelines, which should be based on experimental evidence [4], should be formulated not only as general design principles or low-level and platform-specific recommendations, but should also be added to the actual HCI models that define a user interface.

Research has been published along these lines, aiming to define a model that is adequate for blind users and the task they are to perform. For example, Grammenos et al. [5] suggests that the application interaction needs to be modified to make it more adequate for blind users, without affecting sighted users.

This research, aspiring to model user interfaces for blind people that incorporate accessibility guidelines specified in ISO/UNE standards and specific usability requirements for blind people, falls into this category.

3 User Interface Modelling for Blind People

The HCI model most widely accepted by researchers and designers is the modelling-based definition by [6]. This HCI model involves creating a number of user interface components. They include the task model, the domain model, the dialog model, the presentation model, the platform model and the user model [7]. The *task model* gives a structured representation of what activities the software user may want to perform. The *domain model* describes the syntactic sequence of the interaction and is implemented as a sequence of windows. The *dialog model* describes the interaction between the different objects making up the interface. The *presentation model* describes the user interface's visual appearance. The *platform model* describes the various computer systems that may run a user interface. The *user model* describes the characteristics of the user or user group.

There are several issues that each model has to account for when developing interfaces for blind users. These issues are derived from the usability requirements for blind people described in the introduction: task adequacy, dimensional trade-off, behaviour equivalence, semantic loss avoidance and device independency (Fig. 1).

The **task model** is a formal description of the service the user accesses. It is organized hierarchically and contains information regarding task activation, its preconditions, postconditions, and the actual task action [8].

In the task model, the tasks described should be checked, during problem analysis, for incompatibility with what blind people can do (for the task adequacy requirement). Blind people cannot do activities that require hand-eye coordination, or

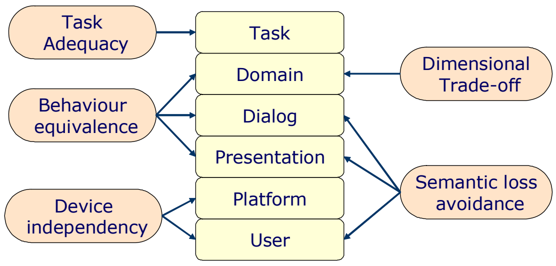


Fig. 1. Influence of blind-user interfacing on HCI models

activities that require simultaneously controlling the state of more than one visual item, because they communicate with the computer serially.

The **domain model** describes the syntactic sequence of human-computer interaction through user interface objects and determines how the set of tasks and actions are ordered [8].

The domain model has to provide a sequence of windows compatible with one-dimensional navigation (for the dimensional trade-off requirement) and has to include speech and Braille output for the transitions between user interface objects (behaviour equivalence requirement). The domain model should define a sequence of windows and window content that is compatible with 1 + 1 navigation (for the dimensional trade-off requirement). Also, the domain model should include the information to be output as speech and Braille in the transition from one window to another (for the behaviour equivalence requirement).

In the first place, the domain model should be represented hierarchically as a series of trees linked by transitions between windows. Each window will be a tree whose root identifies the window in question. The window components, like the title bar, main menu, button bar, etc., will be its children. Each of these nodes will be decomposed into objects (e.g. each of the buttons in the bar) until the objects can be decomposed no further. Any leaf objects whose activation leads to the appearance of a new logic window will have a link to the tree of this new display unit.

One important point when designing the domain model is the tree's depth level for each software window. Because blind users must necessarily navigate by levels within the above tree and cannot "take a look" to find out where to go, the windows should be designed to assure that their navigation tree is not overly complicated. Therefore, the inclusion of several embedded containers is to be avoided.

To cater for the behaviour equivalence requirement, blind users should also be given information about what happens when transitions between tree levels or trees take place. This includes messages about the features of an open item, the type of process that is taking place, the type of objects in the new level or, generally, any information that helps blind users to manage or understand the interface. This is especially relevant in the case of transitions between windows, because blind users could easily get lost. For example, when a message window is opened, the following information should be speech synthesized: window title, message text and active user interface object.

The **dialog model** defines the commands as well as the purpose of each one. Commands are executed via interaction techniques and may produce one or more system responses [9].

The dialog model should enable keyboard-only access, define each user interface object's interaction process, define the access mechanisms to the object's properties (behaviour equivalence requirement), and should incorporate additional interaction techniques to give access to important semantic information (semantic loss avoidance).

First, the dialog model should be developed taking into account that blind users input information using the keyboard or keyboard equivalents (such as speech recognition, Braille keyboards, etc.) only. This is because blind people cannot use pointing devices that require coordinating vision and device movements. Therefore, the type of interaction techniques that this model can use is limited.

It is especially important in this respect for the dialog model to specify that the target platform keyboard usage conventions should be retained. For example, if a key combination is assigned a given function by convention (for example, CTRL + C for copy), then the applications should not attempt to use these combinations for other purposes.

Second, each user interface object's interaction procedure must be defined in line with blind people's capabilities. There should be two chief rules in this interaction procedure: keyboard-based interaction and access through synthesized speech or Braille to the most important attributes of each user interface object.

Therefore, it is necessary to define which of each user interface object's attributes should be able to be queried to be sent to the speech synthesizer and Braille display. In the case of a menu option, for example, access should be given to the text, the mnemonic (underlined letter), associated keyboard accelerator, its state (active or inactive) and, finally, its type: direct command, open dialog or open a submenu.

Finally, and under certain circumstances, alternative interaction techniques need to be taken to perform a given command. This type of actions affects both the dialog and the presentation model (e.g., a musical composition program for blind people should offer special-purpose non-visual mechanisms for editing the score).

The **presentation model** is usually defined as the model that describes the visual appearance of the user interface [10]. This way of looking at the presentation model is good for graphical interfaces, but blind users perceive the interface using other senses, like hearing or touch.

The presentation model should define the speech and Braille output for each user interface object (behaviour equivalence requirement) and should define several detail levels for different user experience levels. It should also guarantee that non-standard objects containing graphical content provide enough semantic information (semantic loss avoidance).

It is essential to take into account the differences there are between the synthesized speech and Braille display outputs, differences that mostly lead to different messages having to be generated for speech and Braille.

For example, information persists in Braille displays until the user takes an action, which is not the case with speech; access to spoken information is sequential (the user has to wait for the voice to reach the fragment of the target information), whereas it is direct in Braille displays (users can move through the displayed information however they like); character attributes (bold, italic, underlined, etc.) can be represented in Braille displays using points 7 and 8 of the 8-point Braille code. These attributes cannot be represented in speech without interrupting the dialog.

Note also that the presentation should make provision for several detail levels. On the one hand, novice blind users will want to receive as much information as possible on each interface element as they learn to use the application. On the other hand, expert users will only want to receive the information that they need to do the job. Our experience has led us to define three detail levels (also termed help levels): beginner, intermediate and advanced.

The user interface designer has to take all this into account to define the speech and Braille presentation of every user interface object. This is a two-step process:

1. Define the presentation of each user interface object's attribute for speech and Braille. For example, object type is represented by full text in speech ("Button") and in abbreviated form in Braille ("BT"). Another example is the presentation of the mnemonic: it is represented as a separate character in speech and as an underlined letter in Braille.
2. Define the *object attributes combination* to generate the output message for each detail level. The designer has to use different sorting techniques for speech and Braille. For instance, the object's name has to appear first in speech to enhance navigation speed.

In doing so, the designer has to take special care to assure that the information sent to the user via speech or Braille is semantically equivalent to what is given visually in the graphical user interface. The goal is to check that users have all the information they need to use the application to do their job.

The **platform model** contains information about the capabilities, constraints and limitations of the target platform [8]. When developing user interfaces for blind people, the platform model is affected by the *device independency* requirement.

For this purpose, the platform model should include speech and Braille output capabilities through standardized APIs, such as but not limited to:

- For speech: text to speech output, stop speech output, automatic spelling and radio code spelling, management of queued speech messages, mechanism for the management of speech progress, changing speech parameters
- For Braille: text to Braille output, providing information about the display characteristics (such as the display size), automatic management of text that is bigger than the Braille display size, management of keys pressed in the Braille display, changing Braille parameters.

Examples of such APIs for speech are SAPI in Microsoft Windows [11] and Java Speech for the Java platform [12].

Finally, the **user model** is a description of the characteristics of an individual user or of a stereotype user group [9]. It is not intended to be a description of the mental state of a user.

The user model has to include configuration parameters related to the semantic loss avoidance and device-independency requirements, such as speech parameters (the device used, speed, tone, volume, etc.) and Braille parameters (the device used, Braille code type, cursor type, presentation of character attributes, etc.) or the detail level of the output messages (for beginner, intermediate and advanced).

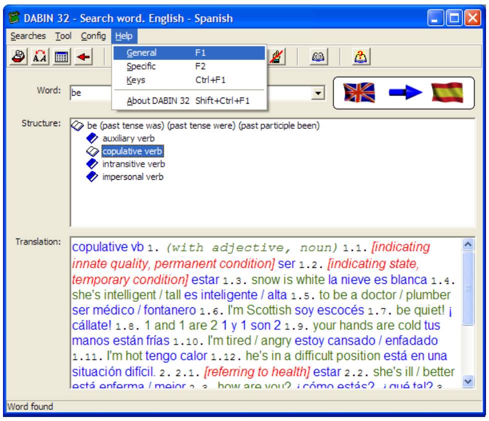
4 A Framework for Blind User Interface Development

Based on the model we have developed a framework for developing user interfaces for blind people, called FBLIND (Framework for BLind user INterface Development). This framework has three main components [13]:

- A set of user interface design guidelines, which is based on software accessibility standards [14] [15] and the experience acquired by our team over the years
- A programming library providing support for Speech and Braille Input and Output (SBIO)
- And an interface development toolkit consisting of automatically adapted user interface objects (BVCL).

Of several applications developed using FBLIND, DABIN, an electronic speaking bilingual software environment for blind or visually impaired people, deserves a special mention. DABIN (Fig. 2) was conceived as an environment housing a range of bilingual dictionaries allowing users to choose the translation source and target languages, as well as the user interface language. It has a dual user interface, and can run word searches, returning all the information about their translation in another language, including grammatical categories, examples, remarks, etc. It now includes a Spanish-English and English-Spanish dictionary, and a Spanish-French and French-Spanish dictionary, and has user interfaces in Spanish, English and French.

Another developed application was the *Proyecto Aprender* (Learn Project) [16]. This system (Fig. 3) is an educational resource targeting children with special educational needs. These children may have an added disability such deafness, blindness and so on. To help blind users to use this application, the FBLIND philosophy was applied to develop the blind user interface.

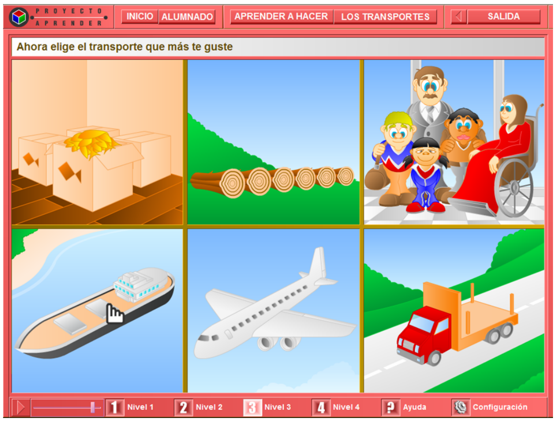

DABIN 32 - Search word. English - Spanish
Searches Tool Config Help
General F1
Specific F2
Keys Ctrl+F1
About DABIN 32 Shift+Ctrl+F1
Word:
Structure:
Translation:
Word found

Word: be

Structure:
be (past tense was) (past tense were) (past participle been)
auxiliary verb
copulative verb
intransitive verb
impersonal verb

Translation:
copulative vb 1. (with adjective, noun) 1.1. [indicating innate quality, permanent condition] ser 1.2. [indicating state, temporary condition] estar 1.3. snow is white la nieve es blanca 1.4. she's intelligent / tall es inteligente / alta 1.5. to be a doctor / plumber ser médico / fontanero 1.6. I'm Scottish soy escocés 1.7. be quiet! ¡cállate! 1.8. 1 and 1 are 2 1 y 1 son 2 1.9. your hands are cold tus manos están frías 1.10. I'm tired / angry estoy cansado / enfadado 1.11. I'm hot tengo calor 1.12. he's in a difficult position está en una situación difícil. 2. 2.1. [referring to health] estar 2.2. she's ill / better está enferma / mejor 2.3. how are you? ¿cómo estás? ¿qué tal? 3

Fig. 2. DABIN application: screenshot



Ahora elige el transporte que más te guste

Fig. 3. *Aprender* application: screenshot

5 Conclusions

Apart from design for all guidelines, developing dual interfaces involves taking into account a set of specific requirements: task adequacy, dimensional trade-off, behaviour equivalence, semantic loss avoidance and device-independent interface. To do this, we designed a user interface model for blind people based on HCI models (the task, domain, dialog, presentation, platform and user models) that takes into account these requirements. As discussed, all six models have to deal with specific issues in order to build user interfaces that are effective, efficient and satisfactory for both sighted and blind users.

To implement the model, we developed a Framework for Blind User Interface Development (FBLIND) that has proved to be very useful for developing dual user interfaces, as it helps developers to focus on important issues, reduces implementation costs and provides device independency.

References

1. Savidis, A., Stephanidis, C.: The I-GET UIMS for Unified User Interface Implementation. User Interfaces for All – Concepts, Methods and Tools, 489–524 (2001)
2. Lemon, G., Faulkner, S.: Making Ajax Work with Screen Readers, `http://juicystudio.com/article/making-ajax-work-with-screen-readers.php`
3. Morley, S.: Design and Evaluation of Non-Visual Information Systems for Blind Users. PhD Dissertation, University of Hertfordshire (1999)
4. Casali, S.P.: A physical skills-based strategy for choosing an appropriate interface method. In: Edwards, A.D.N. (ed.) Extra-ordinary Human Computer Interaction: Interfaces for people with disabilities, pp. 315–342. Cambridge University Press, Cambridge (1995)
5. Grammenos, D., Savidis, A., Georgalis, Y.: Dual educational electronic textbooks: the starlight platform. In: 9th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 107–114. Arizona (2007)

6. Szekely, P., Luo, P., Neches, R.: Beyond Interface Builders: Model-Based Interface Tools. In: InterCHI 1993, pp. 383–390. ACM Press, New York (1993)
7. Myers, B., Hudson, S.E., Pausch, R.: Past, present, and future of user interface software tools. ACM Transactions on Computer-Human Interaction (TOCHI) 7(1), 3–28 (2000)
8. Menkhaus, G., Fischmeister, S.: Dialog Model Clustering for User Interface Adaptation. In: 3rd International Conference on Web Engineering (2003)
9. Puerta, A.: The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development. In: Computer-Aided Design of User Interfaces, pp. 19–36 (1996)
10. Eisenstein, J., Vanderdonckt, J., Puerta, A.: Adapting to Mobile Contexts with User-Interface Modeling. In: Third IEEE Workshop on Mobile Computing Systems and Applications, pp. 83–92 (2000)
11. Microsoft: Welcome to the Microsoft Speech Application SDK, `http://msdn2.microsoft.com/en-us/library/ms986944.aspx`
12. Sun Microsystems: JavaTM Speech API Programmer's Guide, `http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide/`
13. Alonso, F., Fuertes, J.L., González, Á.L., Martínez, L.: A Framework for Blind User Interfacing. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A.I. (eds.) ICCHP 2006. LNCS, vol. 4061, pp. 1031–1038. Springer, Heidelberg (2006)
14. AENOR: Computer applications for people with disabilities. Computer accessibility requirements. Software. Spanish Standard UNE 139802:2003 (2003)
15. ISO: Ergonomics of human-system interaction – Guidance on accessibility for human-computer interfaces. Technical Specification ISO TS 16071 (2003)
16. Fuertes, J.L., González, Á.L., Mariscal, G., Ruiz, C.: Applying a Methodology for Educating Students with Special Needs: A Case Study. In: International Conference on Systems, Computing Sciences and Software Engineering (SCS^2 2007), Springer, Heidelberg (2008)