Abstract UIs as a long-term solution for non-visual access to GUIs

Kris Van Hees, Jan Engelen

Katholieke Universiteit Leuven
Department of Electrical Engineering – ESAT – SCD – DocArch
Kasteelpark Arenberg 10
B-3001 Heverlee-Leuven
Belgium
kris@alchar.org, jan@docarch.be

Abstract

Various approaches to providing blind users with access to graphical user interfaces have been researched extensively in the past 15 years. While graphical user interfaces keep evolving, accessibility is still facing many obstacles that stem from the fact that desktop environments and applications are usually not designed with accessibility in mind. Existing screen readers on MS Windows and X Windows are adequate as short-term solutions, although they generally do not provide access to any arbitrarily chosen application. The higher degree of freedom within the X Windows system further complicates the problem. This paper proposes a long-term solution based on abstract user interface descriptions. Building upon past and current research into user interface description languages, this approach is not only promising for providing blind users with access to graphical user interfaces. It also promotes the "Design-for-All" principle by decoupling presentation and application logic.

1 Introduction

The introduction of graphical user interfaces (GUIs) caused quite a concern within the community of blind users, due to the challenge of providing access to this inherently visual interface (Boyd, Boyd & Vanderheiden, 1990). Until then, blind users gained access to text-based user interfaces by means of screen readers that were capable of inspecting the screen. Since all presented data was pure text, rendering the contents of the interface was relatively easy. GUIs offer a higher degree of flexibility, introducing a wider variety of interaction objects. Amongst all windowing environments, MS Windows has been quite popular in the workplace and at home, and has therefore received quite some attention in view of screen reader development. The relative consistency of the user interface and the interaction objects simplifies the problem. The availability of support for MS Windows has also caused somewhat of a comfort zone.

With the increase in popularity of X Windows-based systems in work and home environments, an additional degree of complexity has emerged. Application developers have access to a whole range of graphical toolkits, such as Athena, GTK, Qt, … While this promotes flexibility and the ability to integrate closely with any of the commonly used desktop environments, it complicates the work needed to provide accessibility. This situation is further complicated by the fact that it is perfectly possible (and often desirable) to run applications developed against a given toolkit in a desktop environment that was developed using a different toolkit. Screen readers therefore cannot depend on specific implementation details.

Past research has indicated that abstracting the user interface offers a high degree of flexibility in rendering for various output modalities. Providing blind users with access to a GUI typically involves providing an auditory and/or tactile representation. It is therefore a good match for using abstract user interface descriptions (AUI). Current approaches used both on MS Windows and X Windows use a combination of graphical toolkit hooks, queries to the application and desktop objects, and scripting to provide accessibility (Weber & Mager, 1996). Unfortunately, this is leading to an "opt-in" situation where applications must explicitly be supported. The proposed approach using an AUI description enables the screen reader to operate on a toolkit-independent definition. It also allows for an implementation as a non-visual rendering agent for the user interface, equivalent to the graphical rendering agents that provide the visual representation.

The remainder of this paper first presents related work on GUI accessibility. The third section focuses on using AUIs at the core of a screen reader, while the fourth section compares the proposed approach with past and current efforts. The fifth section concludes this paper with a description of future work.

2 Related work

Much research has been conducted within the realm of accessibility of GUIs on Unix systems. Mynatt and Weber describe two early approaches: Mercator and GUIB (Mynatt & Weber, 1994). The Mercator project was a research effort at the Georgia Institute of Technology, replacing the GUI with a hierarchical auditory interface. Aside from speech output, it also used short sound fragments to convey iconic information to the user. The GUIB project was a cooperative effort between six European countries, translating the screen contents into a tactile representation.

A more general description of the common approaches towards GUI accessibility can be found in (Gunzenhäuser & Weber, 1994). This early paper also identifies four design issues that are common to non-visual access to GUIs.

The Gnome Accessibility Project aims to provide accessibility to a wide range of disability groups (Haneman & Mulcahy, 2002). The Gnopernicus screen reader lies at the core of the support for blind users, and is currently still under development.

User interfaces are a very important topic within the realm of Human-Computer Interaction. For the purposes of this paper, the UsiXML work done at the Belgian Laboratory of Computer-Human Interaction (BCHI) at the Université Catholique de Louvain is of great importance (Vanderdonckt et al., 2004). The ability to abstract the user interface of applications lies at the core of the methods proposed in this paper.

The similarity between application user interfaces and World Wide Web forms is an important driver as well. Research into specific obstacles that blind user encounter with GUIs (Barnicle, 2000) shows results that are consistent with similar research into web accessibility obstacles (Theofanos & Redish, 2003) and (Pontelli et al., 2002).

Various research projects have investigated the usability of alternative interfaces. Two notable projects are the "virtual sound wall" at the Oldenburg R&D-Institute for Informatics Tools and Systems (Donker, Klante & Gorny, 2002), and the performance analysis of multi-modal interfaces presented in (Vitense, Jacko & Emery, 2002). The rather high cost of the environments involved is a concern, because an average user will typically not be able to afford such sophisticated devices.

The "Fruit" system described in (Kawai, Aida & Saito, 1996) addresses the issue of user interface accessibility as well. This system uses an abstract widget toolkit rather than an AUI description. The application is still written as if a real widget toolkit is being used, while the actual presentation of the user interface is deferred to a device-specific rendering component. As a result, synchronized presentation in multiple modalities is not part of the design, and no features are present to provide accessibility at the windowing environment level.

Another interesting research has been conducted in the Visualisation and Interactive Systems Group of the University of Stuttgart (Rose, Stegmaier, Reina, Weiskopf & Ertl, 2002). Interposing libraries are used to replace the presentation of a user interface. While this is not considered to be a good approach for providing accessibility, it does present an elegant solution for a non-invasive adaptation of a user interface in order to capture widget toolkit function calls for testing purposes. It may also provide a facility to enable legacy applications to use improved or adapted versions of a given user interface toolkit.

Past and current research has resulted in a wide variety of user interface description languages. Many of these are XML-based, and an important subset has been reviewed in (Souchon & Vanderdonckt, 2003). From a universal access and "Design-for-All" perspectives, a comparison was made between four candidate languages in (Trewin, Zimmermann & Vanderheiden, 2003): UIML, XIML, Xforms, and AIAP.

3 AUIs and non-visual access to GUIs

Separating presentation and application logic has been a well-known development paradigm for a long time. Abstract user interfaces (AUIs) take this one step further by eliminating device-specific influences. Visualisation is left to graphical toolkit specific components, generated either programmatically from the AUI, or dynamically by interpreting the AUI at runtime.

3.1 Core of the accessibility solution

In (Edwards, Mynatt & Stockton, 1994) the importance of providing non-visual access to GUIs was placed in direct contrast to providing access to the graphical screen, setting the stage for using alternative representations of the environment rather than trying to interpret the graphical image of windows. This approach implies a decoupling of the user interface from the visual representation, while retaining the interaction semantics. Under the assumption that all applications are developed using a single standard toolkit, it would be sufficient to provide support for screen readers in that one toolkit. Unfortunately, that assumption is generally incorrect. X Windows does not imply the use of any specific graphical toolkit, and it is in fact quite common for users to simultaneously be using applications built upon different ones.

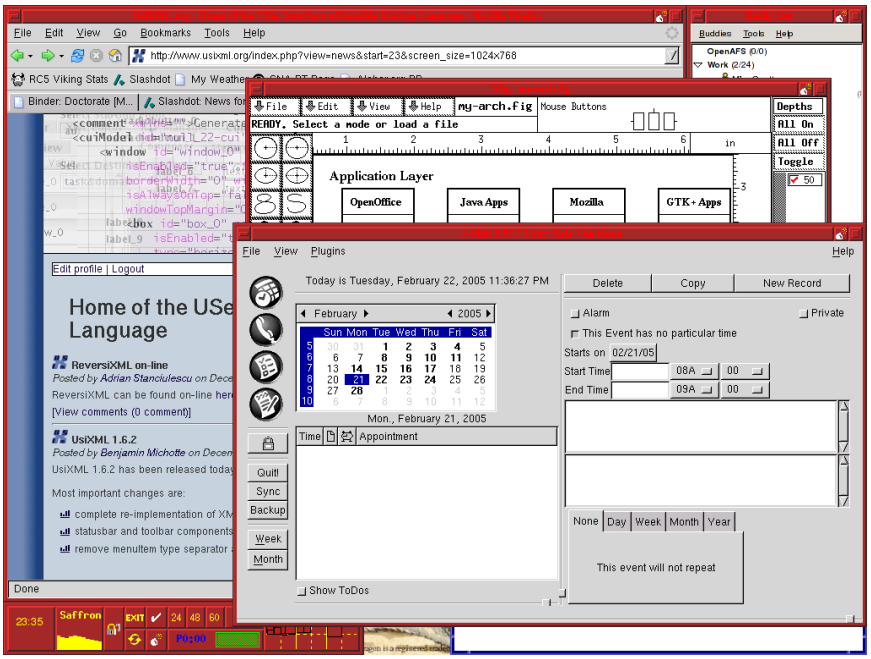

Figure 1: X Windows session showing the simultaneous use of multiple graphical toolkits

Figure 1 shows a fairly typical session on a Unix system, displaying Firefox, Xfig, J-Pilot and GAIM. Firefox and J-Pilot are built upon GTK 1.2, Xfig upon the Athena Widget Set, and GAIM upon GTK 2.0. The bottom of the figure also shows the FVWM button bar. The "Look & Feel" of the graphical interaction objects is quite different, yet sighted users intuitively know how to handle them. All menu bars essentially work the same way, regardless of what they look like. This supports the concept of decoupling presentation and semantics, and lies at the foundation of abstracting the user interface.

AUIs are most commonly used as part of the user interface development process, yielding a final user interface in an appropriate format for the output modality of choice. For web forms this would typically be HTML, while applications may require Java or C source code. This constitutes a compile-time interpretation of the AUI. In order to provide blind users with a non-visual representation, runtime interpretation must also be supported. For the purpose of this paper, the assumption is made that visual rendering of the AUI will take place at runtime as well. While this is not strictly a requirement for the proposed approach, it does greatly simplify the discussion.

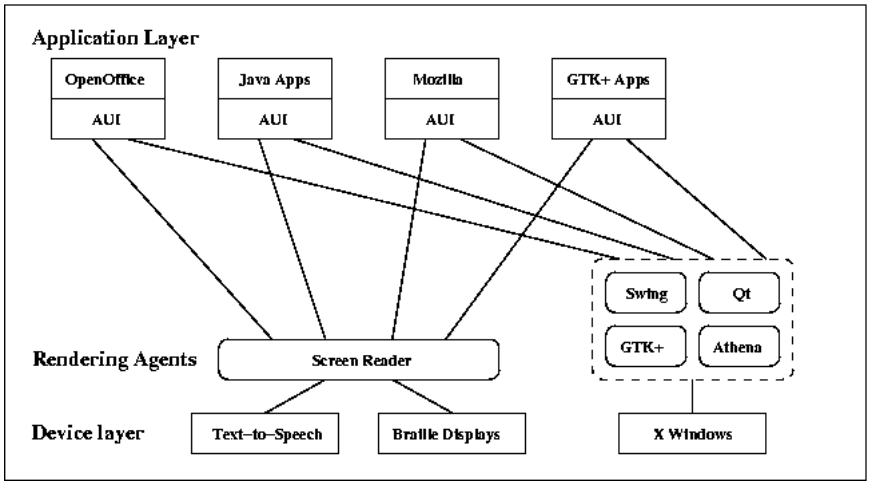

Figure 2: Abstracting the user interface: schematic overview

Figure 2 explains how non-visual access to GUIs can be provided based on abstract user interfaces. The applications are envisioned to have been developed using any available tools, providing an AUI definition in a standard user interface definition language, e.g. UsiXML (Vanderdonckt et al., 2004). The visualisation of the GUI is delegated to widget toolkit specific AUI interpreters, while an alternative AUI interpreter handles the non-visual presentation: the screen reader. Note that for this approach to be successful, the AUI must be capable of representing both data and a description of how to present that data, as suggested in (Trewin, Zimmermann & Vanderheiden, 2003).

3.2 HCI issues for non-visual presentations of GUIs

Rendering of the interface is only one aspect of a successful accessibility solution. Mynatt and Weber discuss four important HCI design issues that need to be addressed as part of any non-visual presentation of GUIs (Mynatt & Weber, 1994):

- **Coherence between visual and non-visual interfaces**
  Collaboration between sighted and blind users requires coherence between the visual and non-visual presentations of the user interface. The mental model of how to interact with an application must be substantially similar to both user groups to allow clear communication about how to accomplish a specific goal. Any user should also be able to observe the actions of another, regardless of what interface is being used by either user.

  The use of AUI interpreters ensures coherence between the presentations because a single source provides the description of the user interface, and the data presented in it. Provided that user interactions are appropriately reflects in the AUI, coherence is guaranteed.
- **Exploration in a non-visual interface**
  Non-visual modalities (auditory and tactile) are limited in their ability to provide information to the user in part due to their largely serial nature, whereas a visual user interface can provide information in parallel in a very efficient way. A screen reader implementation must provide specific non-visual mechanisms to explore the non-visual interface. Given that the GUI is capable of providing information by means of spatial properties of user interface elements (often beyond the scope of a single application), non-visual alternatives must also be provided.

  Within the context of the proposed approach, this means providing access to the application GUIs is not sufficient. Support for the actual windowing environment must also be provided, so potentially important spatial information is not left inaccessible. The screen reader must expose an interaction mode that allows the user to explore the user interface of an application without directly affecting it. This is often called the "review" mode in the screen reader.

- **Conveying graphical information in a non-visual interface**
  The inherently graphical nature of GUIs commonly leads to presenting information in a strictly visual way: icons, object attributes, appearance, … A non-visual presentation must be able to convey relevant aspects of the information in an alternative format.

  Abstracting the user interface into an AUI description implies that visualisation is delegated to AUI interpreters. The presentation of information can therefore no longer be inherently graphical. The non-visual AUI interpreter will render the information in a modality appropriate manner.
- **Interaction in a non-visual interface**
  Interaction in a GUI is often based on visual idioms (clicking buttons, moving sliders, dragging objects, …) whereas a blind user requires specific non-visual forms of interaction.

  It is the responsibility of the screen reader to provide alternative modes of interaction that can be translated into their equivalent visual counterparts. The abstraction of the user interface is therefore not only responsible for the presentation aspect, but also for user input (where needed). An example of this functionality would be translating specific key combinations into mouse operations.

A fifth design issue is provided in (Gunzenhäuser & Weber, 1994):

- **Ease of learning**
  The introduction of non-visual access to GUIs should not be a major obstacle for blind users. The success of the GUI concept depends in part on the intuitive nature of the environment, and on the fact that users can share knowledge easily. Ease of learning can be accomplished by ensuring that the non-visual user interface is sufficiently intuitive to its target group, and that sighted and blind users can share the same mental model of interaction semantics.

The underlying concept for all five HCI design issues is related to how blind users interact best with a computer system. It is therefore a priority for the proposed work to involve the target audience at all stages of research and development.

3.3 Advanced aspects of non-visual access based on AUIs

Any solution for non-visual access to GUIs faces obstacles. Some are implied by the chosen approach, while others are related to the very problem that is being worked on. While a comprehensive list of identified issues is quite lengthy, three important examples illustrate the overall complexity.

3.3.1 Dynamic user interfaces

It is common for user interfaces to contain elements that are somewhat dynamic in nature. Interaction objects may not always be applicable, and are often greyed out to indicate this attribute. This does not alter the composition of the presentation, and therefore does not directly impact non-visual access.

A more disruptive feature involves truly dynamic updates in the user interface. A prime example is a “File” menu on a menubar that displays a list of the last 5 or 10 accessed files. The exact content (or even size) of the menu cannot be determined ahead of time. A possible solution may be the implementation of a feature in the AUI that specifies that this specific content must be queried from an outside source (the application itself). Alternatively, providing a facility for dynamic updates to the AUI description would provide a generic solution to this type of problems.

User interfaces are generally described in an XML-compliant language, providing for a natural hierarchical structure. Allowing the application to update this hierarchy by adding, removing, and updating parts of it ensures that dynamic user interface changes can be supported. The AUI interpreters will be able to pick up these changes and render the new presentations.

3.3.2 Legacy applications

The adoption of AUI-based application development is still a fairly slow moving target. A successful screen reader implementation will therefore be faced with any number of legacy applications that were developed with programmatically defined user interfaces. Although the development of a fully featured, commercial grade screen reader is far beyond the scope of the work proposed in this paper, support for legacy applications will be investigated. Reverse engineering of legacy user interfaces is possible using techniques developed as part of the UsiXML project (Vanderdonckt et al., 2004), and is conceptually equivalent with existing screen reader technology.

3.3.3 So-called "Creative Programming"

By far the biggest obstacle in providing non-visual access to GUIs is "Creative Programming." The flexibility of X Windows empowers software developers to implement very complex user interfaces. In its worst form, a developer may implement his own graphical toolkit. Alternatively, an existing toolkit may be extended with non-compliant widgets that defy heuristics. Creative minds have been known to implement buttons in dialog boxes that "run away" from the mouse pointer once it is within a certain distance.

In summary, it is not feasible to expect a screen reader to be capable of providing non-visual access to each and every application.

4 Comparison with past and current approaches

Two notable past approaches to providing non-visual access to GUIs are described in (Mynatt & Weber, 1994). Mercator replaces the spatial graphical display with a hierarchical auditory interface. A speech synthesis system is added to the standard desktop configuration, and both speech and sound cues are used to convey information to the user. Capturing X-protocol communications and querying toolkit objects, while using the fact that many features of GUIs are related to limitations of the medium, construct an off-screen model. Overlapping windows and clipping occur due to the screen size restrictions, and can therefore be avoided for non-visual access. With the emergence of higher-level graphical toolkits, the capturing of the user interface at the X toolkit level is no longer sufficient. The proposed approach addresses this by operating on an AUI level, independent from the concrete visualisation.

GUIB translates the screen contents into a tactile presentation, retaining spatial organization. A matrix of Braille cells with touch-sensors is used as primary input and output modality, augmented with sound. A virtual screen copy describes the contents of the screen on a lexical level, whereas an off-screen model is used to capture the syntactical structure of the GUI. Because GUIB represents the screen as-is using a 25 by 80 Braille matrix (200 by 160 dots), minimal work is required to transform the hierarchical off-screen model into textual output. While the 1994 paper describes that this was sufficient to represent 640 by 480 pixel screens, current graphical screen technology far surpasses that resolution. Expanding the Braille cell matrix seems impractical, and thereby imposes a limitation to this approach. Retaining the spatial properties of user interfaces is not directly possible with an AUI-based screen reader, and providing facilities for querying the visual rendering agents is planned.

The Archimedes project at the University of Hawaii (formerly at Stanford University) employs a bottom-up approach, capturing an actual image of the graphical screen, and analysing it (Scott & Gingras, 2001). By means of image and optical character recognition techniques, augmented with pattern matching, the image is transformed into an off-screen model that ties into the Total Access System architecture. This system has limited use in an environment where multiple graphical toolkits are available.

An actively developed solution to providing non-visual access to GUIs is the Gnopernicus screen reader, as part of the Gnome Accessibility Architecture (Haneman & Mulcahy, 2002). It supports both speech and Braille output, and aims to provide access to all GTK+2 and Java applications. Figure 3 provides a schematic overview of the Gnome Accessibility Architecture. The sample applications listed at the top of the figure are developed against an accessibility-aware toolkit: the "Access API" for OpenOffice, the "Java Accessibility API" for Java applications, and the "Accessibility ToolKit" for Mozilla and GTK+ applications. Toolkit-specific accessibility bridges provide a standardised interface to the "Assistive Technology – Service Provider Interface". This component is toolkit independent, and allows Gnopernicus to query and interact with GUI interaction objects.

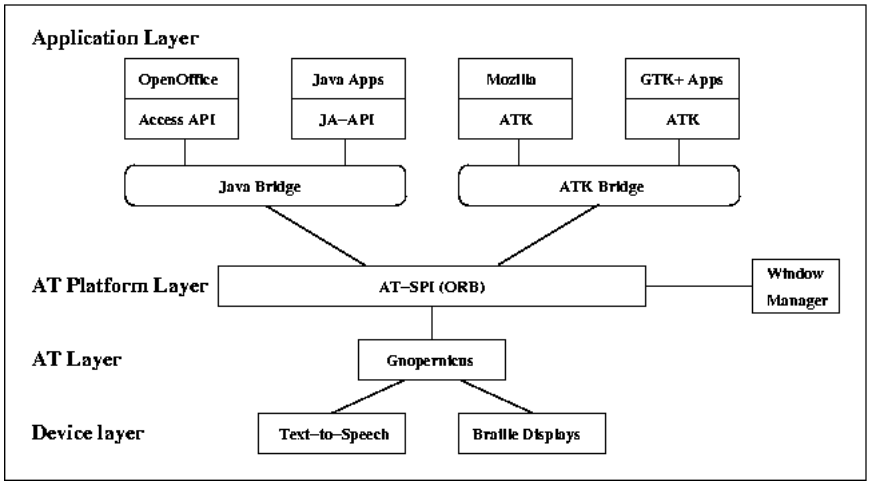

Figure 3: The Gnome Accessibility Architecture: schematic overview

The Gnome Accessibility Architecture is built around the AT-SPI layer, requiring graphical toolkits to implement support for this standard interface. All currently supported toolkits require the application developer to explicitly call functions to provide information that can be queried by assistive technologies providers, such as Gnopernicus. In a posting to the Gnome Accessibility mailing list (July 20th, 2004), Peter Korn stated that Gnome is taking the approach that applications must "opt-in" to accessibility. Unless software does that, it will not work with the screen reader. While the access solution proposed in this paper implies the adoption of AUIs, thereby imposing somewhat of a limitation on supported applications, including techniques to appropriately handle legacy applications is planned because enforcing a specific set of applications upon the users is contradictory to trying to provide a long-term solution for non-visual access to GUIs.

5 Conclusion

This paper presents a long-term solution for providing non-visual access to graphical user interfaces, by means of an abstraction of the user interface. The theory behind the proposed work is built upon extensive research into HCI accessibility issues, abstract user interfaces, and alternative approaches. Only an actual experimental implementation can put it to the test. In the coming months, a very basic visual AUI interpreter will be implemented based on existing widget toolkits. Expanding upon that, a non-visual AUI interpreter will be developed, together with a basic screen reader. Experimental implementations will be presented to blind users throughout the duration of all research and development, to solicit feedback on the techniques used and their effectiveness.

Additional research will be needed in coming months. Appropriate transformations must be defined to translate mostly visual metaphors into non-visual ones. Integration with existing AUI frameworks is also needed, not only because modifications may be required in order to support non-visual access, but also because the proposed approach can only be truly successful if user interface development adopts abstract user interface definitions.

Acknowledgements

The research presented in this paper is part of the author's doctoral work at the Katholieke Universiteit Leuven, Belgium, under supervision by Jan Engelen (ESAT-SCD-Research Group on Document Architectures).

References

Barnicle, K. (2000). Usability testing with screen reading technology in a windows environment. *CUU'00: Proceedings on the 2000 conference on Universal Usability*, 102—109.

Boyd, L. H., Boyd, W. L., Vanderheiden, G. C. (1990). The graphical user interface: crisis, danger and opportunity. *Journal of Visual Impairment and Blindness*, 496—502.

Donker, H., Klante, P., Gorny, P. (2002). The design of auditory user interfaces for blind users. *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction*, 149—156.

Edwards, W. K., Mynatt, E. D., Stockton, K. (1994). Providing access to graphical user interfaces – not graphical screens. *Assets '94: Proceedings of the first annual ACM conference on Assistive technologies*, 47—54.

Gunzenhäuser, R., Weber, G. (1994). Graphical user interfaces for blind people. *13th World Computer Congress 94*, 2, 450—457.

Haneman, B., Mulcahy, M. (2002). The gnome accessibility architecture in detail. Available for download at http://developer.gnome.org/projects/gap/presentations/.

Kawai, S., Aida, H., Saito, T. (1996). Designing interface toolkit with dynamic selectable modality. *Assets '96: Proceedings of the second annual ACM conference on Assistive technologies*, 72—79.

Mynatt, E. D., Weber, G. (1994). Nonvisual presentation of graphical user interfaces: contrasting two approaches. *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, 166—172.

Pontelli, E., Gillan, D., Xiong, W., Saad, E., Gupta, G., Karshmer, A. I. (2002). Navigation of html tables, frames, and xml fragments. *Assets '02: Proceedings of the fifth international ACM conference on Assistive technologies*, 25—32.

Rose, D., Stegmaier, S., Reina, G., Weiskopf, D., Ertl, T. (2002). Non-invasive adaptation of black-box user interfaces. *Proceedings of the Fourth Australian user interface conference on User interfaces 2003*, 18, 19—24.

Scott, N. G., Gingras, I. (2001). The Total Access System. *CHI '01 extended abstracts on Human factors in computing systems*, 13—14.

Souchon, N, Vanderdonckt, J. (2003). A review of XML-compliant user interface description languages. *Proceedings of the 10th international conference on Design, Specification and Verification of Interactive Systems DSV-IS'2003*, 377—391.

Theofanos, M. F., Redish, J. (2003). Bridging the gap: between accessibility and usability. *Interactions*, 10(6), 36—51.

Trewin, S., Zimmermann, G., Vanderheiden, G. (2003). Abstract user interface representations: how well do they support universal access? *CUU '03: Proceedings of the 2003 conference on Universal usability*, 77—84.

Vanderdonckt, J., Limbourg, Q., Michotte, B., Bouillon, L., Trevison, D., Florins, M. (2004). UsiXML: a user interface description language for specifying multimodal user interfaces. *Proceedings of the W3C Workshop on Multimodal Interaction WMI'2004.*

Vitense, H. S., Jacko, J. A., Emery, V. K., (2002). Multimodal feedback: establishing a performance baseline for improved access by individuals with visual impairments. *Assets '02: proceedings of the fifth international ACM conference on Assitive technologies*, 49—56.

Weber, G., Mager, R. (1996). Non-visual user interfaces for X Windows. *Interdisciplinary aspects on computers helping people with special needs: 5th international conference/ICCHP '96*, 459—468.