Accessibility Support with the ACCESS Framework

Michael Heron
University of Dundee
School of Computing
Dundee, DD1 4HN
+44 (0) 1382 385196
michaelheron@computing.dundee.ac.uk

Vicki Hanson
University of Dundee
School of Computing
Dundee, DD1 4HN
+44 (0) 1382 386510
vlh@computing.dundee.ac.uk

Ian Ricketts
University of Dundee
School of Computing
Dundee, DD1 4HN
+44 (0) 1382 384153
ricketts@computing.dundee.ac.uk

ABSTRACT
Equitable access to the digital economy is predicated on the usability of the devices that are used to access electronic goods and services. For novice users with special interaction requirements, current arrangements for enabling accessibility support are sub-optimal. In this paper, we describe the ACCESS Framework, an open-source, plug-in enabled software framework designed to address some of the issues around providing support on the desktop. Through empirical work with older adults, the framework has been shown to provide an understandable, appropriate and effective way to enable accessibility support.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *input devices and strategies, user-centred design*

General Terms
Design, Experimentation, Human Factors

Keywords
Accessibility, older adults, open source

1. INTRODUCTION
In this paper, a new open source software framework, the Accessibility and Cognitive Competence Extended Support System (ACCESS) is described. With this framework user interactions with a system can be analyzed, corrected or supported by small, conceptually simple plug-ins. A total of six plug-ins were deployed for the empirical studies discussed in this paper. These studies focused on the tool within the context of older adults, although it is anticipated that the framework will have applicability within any demographic group where accessibility support is required and computer expertise cannot be assumed.

The research attempted to address three key issues that are problematic for novice users with accessibility and cognitive support needs – that such users or those supporting them often do not know what facilities are available within their operating system, that they often do not know how to enable these facilities if they do, and often lack the confidence to make the changes that they know how to make[2][3].

Digital Engagement '11, November 15–17, 2011, Newcastle, UK.

2. Design of the Framework
The ACCESS Framework incorporates some novel mechanics that must be described to provide the appropriate level of context for the tool. The ACCESS framework acts as a mediator between accessibility plug-ins and the underlying operating system. While the full technical details of the framework are beyond the scope of this paper, in this section we will describe the basics of how the framework operates.

Plug-ins can fill several roles - they can be invokeable, or corrective. Invokeable plug-ins operate as small applications, and are available from a launch-pad within the framework. Corrective plug-ins operate invisibly in the background, analyzing user input for opportunities to make corrective actions.

The framework functions by polling all registered plug-ins every few minutes (in a period known as a **corrective tick**) to see if any register that they wish to perform corrective action. During this tick, the subset of all plug-ins that wish to make a correction is placed into a probabilistically weighted roulette wheel and one is selected at random through spinning this wheel. The selected plug-in is then permitted to make a small correction to the system, such as modifying an appropriate operating system setting, or executing an external application.

User interaction with corrective plug-ins is limited to the expression of preferences through a positive reinforcement and positive punishment model. When a correction is made, the user is presented with a dialog box explaining the change and asking them to register their like or dislike. When they choose **dislike**, the change is reverted and the chance that the plug-in is selected for future corrections is reduced. If they choose **like**, the change is committed and the plug-in is made more likely to be selected for future corrections. In both cases, plug-ins are notified of the direction of user preferences so that they can adjust internal algorithms and thresholds appropriately. If the user selects neither like or dislike, the change is silently committed after several ticks have passed without user feedback, but no changes are made to internal weightings.

The ACCESS framework thus inverts the traditional responsibility of user configuration – plug-ins have responsibility for identifying when a change should be made, and then making that change. All the complexities of user configuration are resolved down to the user indicating their consent or dissent with regards to changes that are made on their behalf.

3. Accessibility Support for Older Adults
Most modern operating systems come with accessibility support built-in. Computer-assisted technologies have the potential for improving the user-experience of people with cognitive, visual and psychomotor issues. However, older users cannot be easily categorized in terms of their need for accessibility or cognitive support. Older users generally do not have one specific issue that

needs to be addressed, but a unique combination of issues. There is often not one obviously serious ailment, but instead a subtle blend of minor ailments, none of which may be considered significant enough in itself to be self-reported as a disability [3].

The level of computer familiarity amongst older users tends to be lower than that of younger adults [4]. Even the accessibility tools that are built into most operating systems require a disproportionate amount of computer knowledge to find and configure [2][4]. Many users would be adequately served by the provided operating system facilities themselves, but these require the user to know the tools exist, how to switch them on, and what the implications are of the changes they are making.

Additionally, the amount of variation between individuals tends to accelerate as age increases. Thus no one specific assistive technology element is likely to suffice, and even if it were it is not assured that an older user would believe they have a need for it. The requirement for individual needs and preferences to be considered when incorporating accessibility is not unique to older users, but is especially relevant to this age group. Solutions that may have been effective at age 60 can become less so at 70 [3]. The needs of an individual in this age group will thus evolve over time more quickly than those with accessibility needs in a younger age bracket. The impact of this dynamic diversity [1] is that even perfectly effective accessibility solutions aimed at supporting a single disability may be ineffective when faced with an older user [3].

It is these issues that inspired the design of the ACCESS framework. While the expressiveness of the framework is limited to the plug-ins it supports, it offers an opportunity to remove the barriers to meaningful accessibility configuration for older adults. Implicit in the design of plug-ins is that the changes made are small and incremental – over time, they find the 'sweet spot' between user requirements and what the operating system can support. This enables subtle blends of configuration options to emerge naturally as a result of many corrections over many plug-ins permitting for a tailored suite of accessibility solutions that meet the dynamic diversity needs of this user group.

4. Plug-Ins

For the studies conducted during this research, six plug-ins were deployed. These are briefly outlined in this section.

Double-Click identified difficulties in registering successful double clicks and adjusted the double click threshold accordingly.

*Missed Click*s identified when users had difficulty in clicking a moving testing target, and enabled pointer precision when this scenario was encountered.

Pointer Size and *Mouse Trails* both identified difficulties in mouse location. The former, when selected to make a correction, successively increased the size of the mouse pointer. The latter enabled and then successively extended the length of mouse trails.

Double Back identified doubling-back behavior of mouse motions (when the course of a motion abruptly changed at the end of an interaction), and adjusted the speed of the mouse as a correction.

Input recorder was deployed during the experimental trials, but never evaluated. It worked invisibly in the background to record an actionable log of user input. It can be invoked to open up a control panel whereby recorded sessions can be loaded and played back, allowing a replay (including operating system interactions) of a user's actions.

5. Study Design

Double-Click was evaluated in a 38-participant, repeated measures study where users first attempted a double-clicking task without the use of the framework, and then again with the framework. This allowed for assessment of the ability of a plug-in to work in isolation to make corrections on a system. The same design was used in a second study to test the remaining plug-ins as part of a suite of five. For both of these studies, users were presented with a computer which had been configured to the extremes of what was permitted by the operating system, and the difference in performance in the with and without conditions was recorded and analysed. Before performing the tasks, participants completed a 21 item, five-point Likert questionnaire regarding their experience and confidence with computers. After the tasks, participants completed a 15 item, five-point Likert questionnaire regarding their perceptions of the framework.

6. Results

The results demonstrated that performance was significantly better with the use of the framework for several of the tasks. Additionally, users reported that they found the framework to be understandable, effective, and that it made changes that they felt were useful. The majority of participants reported that it was a system they would be willing to use on their own systems.

7. Acknowledgements

This research described in this extended abstract was conducted as part of an IBM Open Collaborative Research project between IBM Research and the University of Dundee. Additional funding was provided by a Royal Society Wolfson Merit Award to the second author and by grant RCUK EP/G066019/1 "RCUK Hub: Inclusion in the Digital Economy" Special acknowledgements go to Dr. Norman Alm and Professor Peter Gregor, both of the University of Dundee, for their suggestions, advice and guidance on the tool and its testing.

8. REFERENCES

[1] Gregor, P., Newell, A. F., and Zajicek, M. (2002). Designing for dynamic diversity: interfaces for older people. In *Assets '02: Proceedings of the fifth international ACM conference on Assistive technologies*, pages 151-156, New York, NY, USA. ACM

[2] Hawthorn, D. (2003). How universal is good design for older users? In *CUU '03: Proceedings of the 2003 conference on Universal usability*, (pp. 38–45). New York, NY, USA: ACM.

[3] Pullin, G. and Newell, A. (2007). Focussing on Extra-Ordinary users. In *Universal Acess in Human Computer Interaction. Coping with Diversity*, pages 253-262.

[4] Trewin, S. (2000). Configuration agents, control and privacy. In *CUU '00: Proceedings on the 2000 conference on Universal Usability*, (pp. 9–16). New York, NY, USA: ACM.

[5] Trewin, S. (2004). Automating accessibility: the dynamic keyboard. In *Assets '04: Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility*, (pp. 71–78). New York, NY, USA: ACM.