

Inclusive Design for Ordinary Users in Extraordinary Circumstances

Simeon Keates

IT University of Copenhagen
Rued Langgaards Vej 7, DK-2300 Copenhagen S
skea@itu.dk

Abstract. Universal access is commonly interpreted as focusing on designing for users with atypical requirements – specifically users with disabilities or older adults. However, universal access is also about providing access to users in all situations and circumstances, including those that place extraordinary burdens on the users. This paper examines the design of a user interface (UI) for use in an airport environment and explains how the lessons learned from designing for users with disabilities in particular have been applied in this new context. The paper further describes a series of experiments that were performed to demonstrate the usability of the new interface and also compare the efficacy of three different input strategies developed for the new UI. The most efficient method of input was a strategy of combined keyboard shortcuts offering access to the full functionality of the UI.

1 Introduction

A quick look through the technical programmes of many conferences addressing the thematic area of universal access shows that historically the overwhelming majority of papers address design for disabled users or design for older adults [4]. However, it has long been claimed that the techniques developed to design for such extraordinary users in ordinary circumstances could also be applied to ordinary users in extraordinary circumstances [3]. Indeed, many funding applications often feature this very claim.

Recently, more research has been performed on examining the role of innovative circumstances, or contexts, of use. For example, designing for ambient intelligence is one such area that has received much attention in the past few years [1].

This paper describes the development of a new user interface (UI) for use in an airport environment. The specific application was a Departure Control System (DCS).

Airports are very challenging environments in which to use computers, especially since many computers there are often old and, most importantly, do not have mice. While many UIs exist for use without mice, such as traditional command-line interfaces, the DCS UI was developed as part of an overall suite of applications that required a coherent branded appearance. Many of the other applications were to be used in a more typical office environment and needed to support typical mouse and keyboard input and resemble a traditional graphical user interface (GUI).

Thus the DCS UI needed to have the look and feel of a typical GUI application and support mouse input, but actually be optimized for keyboard-only access. These design requirements represented an interesting challenge for the design team; a challenge that was resolved by examining design techniques used for users who cannot use a mouse, specifically blind users and those with severe motor impairments. In other words, methods developed for designing for “extraordinary” users were applied to a UI for “ordinary” users in “extraordinary” circumstances [3].

1.1 The Airport Environment

The airport environment itself is a very demanding one. Airports are often very noisy places, crowded and with unequal and often poor lighting. Airline staff are under constant time pressure to process passengers as swiftly as possible. In the event of a last minute cancellation or other service disruption, the members of airline staff need to have accurate and reliable information made available to them in a timely fashion to keep passengers informed and defuse any potential confrontation.

Airlines often do not have any control over the computer equipment available to them. While most airlines may invest in purchasing their own equipment for their major hubs, smaller airports will almost certainly use common use platforms. These are computer facilities that are provided by a subcontractor through the airport to the airlines.

The common use platform machines are typically Linux or Windows NT based and often support only 1024×768 CRT displays. Some are equipped with hand-scanners for processing 1-D and 2-D barcodes, but those scanners are notoriously unreliable.

Most crucially, they often are not equipped with mice.

1.2 The Departure Control System

Airline operations within an airport are very complex and diverse. The most obvious features to most people are those that they interact with as passengers – specifically check-in and boarding. While these are, in themselves, large design spaces, they represent only a fraction of what is often going on. A complete Departure Control System (DCS) needs to include the following functionality:

- Check in – identify passengers and whether they are travelling as part of a group, ensure that the passengers are correctly ticketed and have the necessary documentation to be eligible to travel, assign the passengers to seats or to the standby list, process their baggage (see below), collect any outstanding payments and issue boarding passes
- Boarding – prepare the flight for departure, perform all necessary security checks, ensure that all passengers that are eligible to fly board the aircraft and that all ticketed passengers that are denied boarding (for a variety of possible reasons) are found seats on alternative flights
- Baggage handling and tracking – ensure that all bags are correctly entered into the system, have tags issued and are tracked correctly, especially for lost luggage claims

- Catering – ensure that the correct number and type of special meals are onboard each aircraft
- Irregular operations – facilitate a smooth transfer of passengers from flights that have experienced sudden difficulties (e.g. cancelled because of adverse weather conditions) to other available flights
- Standbys and denied boardings – add or remove passengers from a particular flight prior to boarding
- Special services – ensure that passengers with particular special service requirements, such as wheelchair, are correctly identified and that the appropriate service is made available
- Flight manifests – ensure that the passenger and baggage manifests are complete for customs and immigration clearance

1.3 Summary

In summary, the airline personnel have to use machines running old technology that can be temperamental at times. They are under significant time constraints and need to have a high productivity rate (throughput). These are issues that can be addressed by traditional user-centred / usability design practices. The specific issue of interest for this paper is the absence of computer mice. The hypothesis to be examined here is that by drawing on the experiences of designing for users who cannot use a mouse because of severe motor impairments, a better design for nominally able-bodied users who do not have access to a mouse can be created.

2 The DCS Design Approach

As a whole, the DCS design approach has followed an agile methodology. Rather than developing a complete detailed design specification prior to the coding of the application, the design and coding were both tackled at the same time. Thus, fast design creation and iteration was required to ensure a mature enough design was available to the coders in a just-in-time fashion.

To achieve the necessary design speed, it was important to establish a close working relationship with the client's subject matter experts. Weekly brainstorming sessions were set-up, supplemented by daily telephone conversations. Wireframe prototypes were developed in PowerPoint as a rapid prototyping tool – the ability to add “Action settings” to on-screen elements (hyperlinking to other slides in the deck) allowed a good simulation of actual use of the interface in a very short development time, without having to wait for the code-base to be fully developed and implemented.

The design of the UI had three principal goals within this overall approach:

1. To ensure that all of the necessary functionality was made available to the users
2. To ensure that the users could accomplish their daily tasks as time-efficiently as possible
3. To minimise training requirements for airport employees (specifically less than one day of training)

To achieve these goals, and the implicit sub-goals, within the development timeframe required a clear design vision.

2.1 The Design Vision

The other applications in the overall product suite were developed very much as Web 2.0 applications, using typical Web 2.0 metaphors and features. These were appropriate given the nature of the tasks.

The design of DCS was fundamentally different. For instance, forward and back buttons are meaningless in a DCS context. When a user performs an action, for example changing a passenger's seat allocation, that action is committed immediately, fundamentally changing the state of the DCS system. While it may be possible to back out that action, it is not guaranteed. For example, the passenger's original seat may have been allocated to another passenger in the time taken for the check in agent to undo the seat exchange. Close to the close of check-in for flight, this type of situation becomes increasingly common as the aeroplane is becoming full.

Consequently, the design had to support two correlated properties:

- Fast throughput for the users
- Minimised risk of user errors

An effective method for achieving both of these properties is to ensure that the users feel confident in how the application will respond to their input commands. Before they press a button, they should be sure what the outcome of that button press would be.

Thus, rather than considering the design of DCS as a typical Web application, the metaphor used was that of an ATM. The analogy of an ATM was chosen because, for example, not all actions offered by an ATM are easily countermanded and mice are not typically used. The user has to feel totally in control of how the system is operating – otherwise they would not trust their money to it. Even the visual appearance would be somewhat similar, with the need to keep pages lightweight (in size) and thus comparatively simple.

Further, the process of interaction in DCS was modelled as a series of flows – the check-in flow, the boarding flow, etc. – with self-contained, discrete modules of additional information (such as advanced passenger information, special service requests or limited release baggage) added orthogonally to the main basic flows. Each flow was modelled with its own progress bar / breadcrumb trail, clearly indicating the current stage in the overall task flow.

Finally, the design needed to visually resemble a typical GUI application, but be optimised for keyboard-only access.

3 Designing for Keyboard-Only Access

The standard approach to designing keyboard-accessible web pages is to follow the Web Content Authoring Guidelines 1.0 (WCAG 1.0) from the Web Accessibility Initiative (WAI) of the World-Wide Web Consortium (W3C) [5] - specifically Guideline 9 – “Design for device-independence” has the following checkpoints:

“9.4 Create a logical tab order through links, form controls, and objects. [Priority 3] For example, in HTML, specify tab order via the “tabindex” attribute or ensure a logical page design.

...

9.5 Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls.”

Further, WCAG 1.0 is due to be superseded by WCAG 2.0 [6], which states:

“Guideline 2.1 Keyboard Accessible: Make all functionality available from a keyboard.

2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)

...

Note 2: This does not forbid and should not discourage providing mouse input or other input methods in addition to keyboard operation.

2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)”

The screenshot shows a web browser window titled "Polarix ResClient" with a "Departure Control" tab. The page has a menu bar with "File", "View", and "Help". Below the menu is a navigation bar with "DCS" and a series of links: "Find", "Confirm", "F2 Status", "F3 Docs", "F4 Seats", "F5 Baggage", "F6 Payment", and "F7 Printing". The "Find" link is highlighted with a red arrow. The main content area is titled "Find primary booking" and contains a form with the following fields and controls:

- Charges/refunds:** Total: \$0.00
- Flight summary:** Flight #, Date, From, To, Boarding, Departure, Arrival.
- Flight occupancy:** Cabin (J, Y), Capacity, Booked, Ch. in, Avail, Standby, INF.
- Form fields:**
 - 1. Last name: Shakespeare
 - First name (optional):
 - Date: ddmmyyyy
 - Flight: - or -
 - Leaving from: Boston BOS
 - + Going to: Vancouver YVR
 - 2. Booking reference
 - 3. Frequent flyer
 - 4. Ticket number
 - 5. Credit card number
 - 6. Phone number (Country, Area, Number)
- Buttons:**
 - (F9) Passenger details...
 - (F9) Express check-in...
 - Express booking
 - View flight status
 - Search for booking... (with a red arrow icon)

The status bar at the bottom shows "Ready".

Fig. 1. The check-in search page from DCS

Taking these as a starting point, it can be seen that TAB key navigation between page elements should be supported wherever possible and keyboard shortcuts implemented for commonly used features.

The first design approach for DCS was to consider that the TAB key could be used to navigate between fields and keyboard shortcuts added to the buttons. This was the approach adopted for the other applications in the overall software suite. However, while such an approach yields a design that is accessible without the use of a mouse, it would still not provide the levels of keyboard optimisation required in the airports. More innovative thinking was required.

Looking at the initial search page for the DCS check-in application for finding a passenger who wishes to check-in to a flight (Figure 1) there are 6 possible search fields, some of which have multiple text entry fields. Using the TAB key, it would take 11 TAB key presses to reach the first text entry field for a phone number search. A faster method of navigation was required for the levels of throughput demanded in an airport.

3.1 Optimising for Keyboard-Only Input – The DCS Solution

Clearly to obtain sufficiently high levels of throughput, more radical changes to the keyboard input paradigm were required. A range of additions and changes were made to the typical “keyboard-isation” of Web pages [2]. These included:

- **Supporting the use of arrow keys between text entry boxes.** In the case of Figure 2, using the down arrow key would require 7 keystrokes to reach the telephone search fields, rather than the 11 for the TAB key (a 36% reduction in keystrokes). In DCS, all screens allowed the use of arrow keys to move between text entry fields.
- **Supporting the use of ALT+[number] to move to text entry boxes.** Looking closely at Figure 2, it can be seen that each main search field has been assigned a number and that number is underlined – the standard labelling technique for ALT-enabled keyboard shortcuts. In this case, pressing “ALT+6” jumps the focus from the first text entry field to the country code entry field for telephone searches. Thus the 11 TAB keystrokes required to reach this field can be accomplished by 1 compound keystroke (a 91% reduction).
- **Enabling direct selection of list items.** On many pages within DCS, a list is presented from which items have to be selected. Common examples include a list of passengers on the same booking, checked (i.e. hold) baggage and fees to be paid. Using the TAB key would require the addition of check-boxes next to each list item. The user would have to TAB to the check box and press either SPACE or ENTER to change the check box state. In DCS, all items in such lists are either numbered or lettered. Pressing the appropriate key (when keyboard focus is not in a text entry field) toggles the selection/de-selection of the respective item. Thus pressing “2” twice will select and then de-select the second item in an on-screen list.
- **Constraining the TAB key.** If coded like a typical Web page, once the TAB key has been used to move the keyboard focus to the “Number” text entry box in the Telephone Number search field in Figure 2, then pressing the TAB key again will move the focus to the buttons at the bottom of the screen. Continuing to press TAB will move the focus to the menus at the top of the screen and through the different tabs, adding approximately 11 TAB keystrokes to return to the first text entry box.

Since all of those other elements have their own keyboard shortcuts, the TAB order can be constrained to just the text entry boxes, meaning just a single TAB key-stroke will move the focus from the last text entry box on the page to the first.

- **Using INS to toggle between direct selection mode and text entry mode.** The direct selection mode works only if the keyboard focus is not on a text entry box – otherwise pressing a number puts that number into a text entry field rather than selecting/de-selecting an item from a list. Moving into and out of a text entry field can be achieved by simply TAB-ing. However, to streamline the process, DCS supported the use of the INS key to jump immediately to the first text entry box on the page, irrespective of where the TAB may be on the page.
- **Adding function keys to the “flow” (breadcrumb trail).** Typically breadcrumb trails have to be traversed in full to reach a particular link. In DCS, each stage of the “flow” (the specialised version of a standard breadcrumb trail) is associated to a function key. Thus pressing the appropriate function key jumps straight to the stage of the flow.
- **Adding a default button to each page.** While each button in DCS includes its own ALT+[underlined_letter] shortcut, the button most likely to be used on that page (most commonly the “Continue”-type button) is shaded in green and can be operated by pressing ENTER, as well as ALT+C (in the “Continue” case). This is very similar operation to Mac OSX button behaviour. This may look like a negligible performance improvement (a single keystroke instead of a compound one), but proved to be very popular with users.

4 Evaluating the DCS Check in UI

The DCS check in user interface was the first component to be coded and evaluated. A 2-stage evaluation process was adopted:

1. Ensure the functional completeness of the UI and its learnability
2. Evaluate the efficiency of the keyboard input paradigm.

4.1 Validating the Functionality of DCS

The DCS user interface was assessed through a series of user trials over 4 days with 10 experienced airport personnel. Each user was asked to perform 12 typical and atypical check-ins. These ranged from domestic passengers with no bags, to a family of four where one passenger gets called away on an emergency and even a passenger that arrives at the airport for an international flight, only to discover that he cannot find his passport. All of the users were given 20 minutes of training – consisting of:

- A brief overview of DCS and the keyboard navigation paradigm;
- A screen-by-screen overview of the UI; and,
- A sample check-in to perform.

All of the users were able to complete the tasks that they attempted (not all had enough time to attempt all tasks) and found the keyboard navigation to be intuitive and easy to learn and use. The only exception was the use of the INS key, which was

effective, but unpopular. While all of the users understood its use, they felt that it did not feel quite right. The use of INS was removed and now TAB is used to move into and out of text entry boxes.

The user comments were strongly favourable, with a unanimous acceptance that the required functionality had been included and also that the taskflows implemented were appropriate and complete.

4.2 Validating the Keyboard Input Paradigm

The second stage of DCS evaluation involved comparing the efficacy of the keyboard input paradigm, specifically the time to complete typical check in tasks. Two tasks were selected:

- 1. Search for a booking using the phone number associated with the booking
- 2. Complete a domestic check in for 2 passengers with 3 bags

Additionally, 3 input methods were chosen for comparison:

- 1. TAB key only (no mouse use)
- 2. Mouse (and keyboard for data entry)
- 3. All keyboard shortcuts (no mouse use)
- 4. Arrow keys (search task only)

All users performed the search task 4 times per input method. The domestic check in was completed once per interaction method. 4 users, all familiar with the new DCS UI, participated in the evaluation and the interaction methods were randomly presented to avoid any order effects. However, all users performed the search task first and then the complete domestic check in task.

4.3 Search Task Results

The results for the search task are shown in Table 1.

Table 1. The mean times taken to complete the search task for the 4 different input methods. Also shown is the % difference from the mouse time. Positive % means slower input, negative % means faster.

Input method	Mean time to complete (s)	% difference from Mouse
TAB key	11.0	+8.8%
Arrow keys	11.5	+12.7%
Mouse	10.2	(-)
Shortcuts	8.3	-18.0%

It can be seen that using the TAB key only for navigation is 8.8% slower than using the mouse. Using the arrow keys is slower still, 12.7% slower than using the mouse. This is somewhat surprising since the critical path (i.e. the shortest possible route to complete the task) is 4 key presses fewer for the arrow keys than for the TAB key (11 arrow key presses instead of 15 TAB key presses), since the two right hand text entry boxes can be skipped using the down arrow key (see Figure 1). The most

likely explanation for this difference is that use of the arrow keys for this type of navigation is no longer common and is much more unfamiliar now to GUI users than the use of the TAB key.

The fastest method of input was the full use of the shortcuts. The critical path here was:

ALT+6, [country code], TAB or RIGHT ARROW, [area code],
TAB or RIGHT ARROW, [phone number], ENTER

This gives a total of 4 key presses that are non-data entry (cf. 13 and 11 respectively for the TAB and ARROW keys). Here the unfamiliarity of the ALT key shortcut is more than offset by the reduction in total number of key presses required.

4.4 Domestic Check in Results

Table 2 shows the results for the domestic check in results for the 3 different input methods compared.

Table 2. The mean times taken to complete the domestic check in task for the 3 different input methods. Also shown is the % difference from the mouse time. Positive % means slower input, negative % means faster.

Input method	Mean time to complete (s)	% difference from Mouse
TAB key	111.8	+29.7%
Mouse	86.2	(-)
Shortcuts	58.2	-32.5%

Table 2 shows that using the TAB key is almost 30% slower than using the mouse, whereas the use of the full range of keyboard shortcuts is over 30% faster. The difference between the TAB key input method and the full range of shortcuts is even larger – with the TAB key taking almost twice as long to complete the task.

5 Conclusions

The results shown in Tables 1 and 2 clearly demonstrate that relying only on the TAB key to provide principal keyboard-only navigation leads to interaction times that are between 9 and 30% slower than using a mouse and keyboard combination. This result is not unexpected, but shows that if designers take the easy route of only supporting this and think that the final UI is “accessible” then they are not doing all that they can for the users.

The benefits of taking a thorough user-centered, information architecture approach to the design process can be seen from the overall time savings for the full range of keyboard shortcuts. By supporting all the keyboard input techniques described in this paper, it has been shown that keyboard only entry can be up to 32.5% faster than using a mouse and keyboard combination. This is a significant improvement in performance, allowing each check in agent to process almost 3 groups of passengers in

the time it would take to process 2 groups using the mouse. Such an improvement is of great financial value to an airline.

The overall design approach of looking at techniques developed for designing for accessibility (specifically designing for users with vision or motor impairments) was effective in suggesting methods of enabling keyboard-only access. This case study demonstrates that those methods can be applied when designing for ordinary users in extraordinary circumstances.

References

1. Adams, R., Granic, A., Keates, S.: Are ambient intelligent applications universally accessible? In: Karwowski, W., Salvendy, G. (eds.) *Proceedings of AHFE International*, Las Vegas, NV, July 14-17 (2008)
2. Keates, S.: Designing for accessibility: Extending to ordinary users in extraordinary circumstances. In: Karwowski, W., Salvendy, G. (eds.) *Proceedings of AHFE International*, Las Vegas, NV, July 14-17 (2008)
3. Newell, A.F.: Extra-ordinary Human Computer Operation. In: Edwards, A.D.N. (ed.) *Extra-ordinary Human-Computer Interactions*. Cambridge University Press, Cambridge (1995)
4. Stephanidis, C. (ed.) *Proceedings of 4th International Conference on Universal Access and Human computer Interaction*, Beijing China, July 22-27 (2007)
5. W3C. Web Content Accessibility Guidelines 1.0 (1999),
<http://www.w3.org/TR/WCAG10/> (accessed: February 20, 2009)
6. W3C. Web Content Accessibility Guidelines (WCAG) 2.0 (2009),
<http://www.w3.org/TR/WCAG20/> (accessed: February 20, 2009)