THE APPLICATION OF A METHOD FOR INTEGRATING NON-SPEECH AUDIO INTO HUMAN-COMPUTER INTERFACES

Stephen A. Brewster[1], Peter C. Wright[2] and Alistair D. N. Edwards[2]

[1]VTT Information Technology,
Tekniikantie 4 B,
P.O. Box 1203,
FIN-02044 VTT, Finland
Tel.: +358 0 456 4311
sab@hemuli.tte.vtt.fi

[2]Department of Computer Science,
University of York,
Heslington,
York, Y01 5DD, UK.
Tel.: +44 904 432775
[pcw, alistair]@minster.york.ac.uk

ABSTRACT

This paper describes the application of a structured method for integrating non-speech sound into graphical interfaces. The method analyses interactions in terms of event, status and mode information. It then categorises this information in terms of the feedback needed to present it. This is then combined with guidelines for creating sounds to generate the auditory feedback required. As an example, the method is applied to a scrollbar. This sonically-enhanced scrollbar is then experimentally evaluated to see if the auditory enhancements are effective. The results show that the new scrollbar reduced the time taken to perform certain tasks, it reduced the time taken to recover from errors, it reduced mental workload and participants preferred it to the standard graphical scrollbar.

KEYWORDS

Auditory interfaces, earcons, sonically-enhanced scrollbar, sonification, multimodal interaction

INTRODUCTION

The combination of graphical and auditory information at the human-computer interface is a natural step forward. In everyday life both senses combine to give us complementary information about the world. The visual system gives us detailed data about a small area of focus whereas the auditory system provides general data from all around, alerting us to things outside our peripheral vision. The advantages offered by the combination of these different senses can be brought to the human-computer interface. Dannenberg & Blattner ([6], pp xviii-xix) discuss some of the advantages of using this approach in multimedia/multimodal computer systems:

> "People communicate more effectively through multiple channels. ... Music and other sound in film or drama can be used to communicate aspects of the plot or situation that are not verbalised by the actors. Ancient drama used a chorus and musicians to put the action into its proper setting without interfering with the plot. Similarly, non-speech audio messages can communicate to the computer user without interfering with an application".

Currently, almost all information presented by computers is visual. A multimodal interface that integrated information output to both senses could capitalise on the interdependence between them and present information in the most efficient way possible.

How then should the information be apportioned to each of the senses? Sounds can do more than simply indicate errors or supply redundant feedback for what is already available on the graphical display. They should be used to present information that is not currently displayed (give more information) or present existing information in a more effective way so that users can deal with it more efficiently.

Alty & McCartney [2] have begun to consider this problem in process control environments. They wanted to create a multimedia process control system that would choose the appropriate modality (from those available) for presenting information to the plant operator. A resource manager was to be used for this. In such a system there is much information that must be presented and the appropriate modality may not always be available because, for example, it is being used for other output at the same time. Alty & McCartney suggest that alternative media could then be

used. Unfortunately, at the present time there is no method for deciding which is the best way to present the information. They say (p 10):

> "...almost nothing is known about what constitutes the successful use of multiple media and modes of communication with the user. Hence, it is extremely difficult to specify any heuristics or constraints which may be successfully employed to drive a resource management system".

The abilities of their resource manager were limited for this reason. Alty & McCartney suggested that further research was needed on rules for combining media and modes before the system could operate without the system designer specifying what modalities were needed. The research described in this paper aims to deal with some of these issues by providing a technique that suggests where and what sound could effectively be used in the interface.

The structured use of sound at the interface is still in its infancy and so sound is often used in an *ad hoc* way by individual designers. Gaver [20, 21] has used a more principled approach, for example in the SonicFinder he used everyday sounds in ways suggested by the natural environment. However, in an interface there are many situations where there are no natural equivalents in the everyday world. The SonicFinder also used sound redundantly with graphics. This proved to be effective but it is suggested here that sound can do more.

Although there have been effective systems using sound (for example, Gaver's ARKola system [22]) some have had little success, for example Barfield, Rosenberg & Levasseur and Portigal [4, 37]. Barfield *et al.* carried out experiments where they used earcons to aid navigation through a menu hierarchy. They say (p 102): "...the following study was done to determine if using sound to represent depth within the menu structure would assist users in recalling the level of a particular menu item". The earcons they used were very simple, just decreasing in pitch as a subject moved down the hierarchy. The sounds lasted half a second. They describe them thus (p 104): "...the tones were played with a harpsichord sound and started in the fifth octave of E corresponding to the main or top level of the menu and descended through B of the fourth octave".

These sounds did not fully exploit all the advantages offered by earcons (for example, they used neither rhythm nor timbre and did not exploit the highly structured nature of earcons) and they did not improve user performance in the task. If better earcons had been used then a performance increase may have been found. This work shows that there is a need for a set of guidelines for the creation of effective earcons. If earcons are created without care then they may be ineffective. Another reason for the lack of success of the earcons in this experiment was that they might not have been used in the best places in the interface. There are no rules about where to use sounds, the choice is up to individual designers and so mistakes can occur.

Portigal conducted an experiment to investigate the extraction of document structure using graphics, earcons and a combination of both. His results showed that in the graphical and combination conditions the results were the same but more effort was required with the combination. The purely auditory condition was the worst of all. Again, there are two possible explanations for this. The first is that the sounds themselves might have been ineffective, the second that they might not have been used to present the most appropriate type of information. Our aim is to help designers avoid these problems by providing a structured method by which they can add effective sounds to their interfaces. A method is needed to allow designers to create effective earcons and to find situations in an interface where sound might be useful. Prior to the work described here, there was no such method. It should provide for a clear, consistent and effective use of non-speech audio across the interface. Designers will then have a technique for identifying where sound would be useful and for using it in a more structured way.

One question that might be asked is: Why use sound to present the extra information? A graphical method could be used instead. The drawback with this is that it puts a greater load on the visual channel. Furthermore, sound has certain advantages. For example, it can be heard from all around, it does not disrupt the user's visual attention and it can alert the user to changes very effectively. It is for these reasons that we suggest that sound be used to enhance the user interface (for a more detailed discussion see [9]).

This paper has three parts. It first described the two components of the structured method for integrating non-speech audio into human-computer interfaces: The analysis technique for finding where to add sound and guidelines for creating the sounds. The next part of the paper shows the method applied to a scrollbar. Problems with the scrollbar are identified and sounds designed to overcome them. The final part of the paper describes an experiment to test the effectiveness of the method. It shows an experiment to test a sonically-enhanced scrollbar based on the design described in the two previous sections.

WHERE TO USE SOUND AT THE INTERFACE

There are potentially many ways that sound could be used in an interface. As Alty & McCartney have shown there is no existing set of rules that says what information is best presented in sound. The approach taken in this paper is to take one type of information and investigate it in detail. The research described here proposes using sound to present

information that is hidden from the user in the interface because when information is hidden errors occur. The structured method described identifies *where* hidden information exists and shows how sound can be used to present it.

Casner & Lewis [41] suggest hidden information is a problem because (p 197): "Understanding how to use a computer often requires knowledge of hidden events: Things which happen as a result of users actions but which produce no immediate perceptible effect". Considering the problems of hidden information is similar to the approach taken by Blattner, Papp & Glinert [7] when adding sound to computerised maps. They suggested that information could become hidden because of visual clutter: Only so much information could be displayed before the underlying map was obscured. If additional information was to be displayed on a map, space must be allocated for it. Eventually a saturation point will be reached where interference with the existing graphics and text cancels out any benefit from adding more information. Blattner *et al.* suggested that sound could be used to avoid these problems and make explicit the hidden information. We suggest information may be hidden in an interface for a number of reasons:

Information is not available: It may not be available on the display due to hardware limitations such as lack of CPU power or screen size.

Information is difficult to access: It may be available but be difficult to get at. For example, to get file size and creation date information on the Macintosh a dialogue box must be called up.

Too much information: It can be hidden if it scrolls by too fast because the user's visual system is overloaded. As Blattner *et al.* [7] discussed, presenting too much information in a small area may effectively cause some of it to be hidden.

Small area of visual focus: It can be hidden because it is outside the small area of focus of the visual system. The user may not be looking at the right place at the right time to see what is displayed.

Screen space: There is a trade-off between screen space and the salience of status information [39, 40]. Scott & Findlay [40] showed that by increasing the amount of screen space taken up by the status information they could decrease the time to perform an editing task. In their most salient feedback case two complete rows of the screen were taken up: One at the top the other at the bottom. This proved to be an effective way of communicating the status information but at the cost of a great deal of screen space that could otherwise have been used for user data.

Modes: A mode is a state within a system in which a certain interpretation is placed on information (for example in one mode characters typed into an editor may appear on the screen, in another they may be interpreted as commands). It is often the case that the details of this interpretation are hidden from the user and mode errors result [41]. As Thimbleby ([44], p 228) says: "Typically, the user finds modes difficult because much important information is *hidden*, perhaps forgotten or not noticed by the user". A much more detailed discussion of the problems associated with modes is given below.

How can sound help with these problems? If information is hidden because of hardware limitations such as screen size then sound could be used to overcome this. Almost all computers have the necessary sound output hardware built-in and for most of the time it is lying idle. Information that must be presented visually could be displayed on the screen whilst other information could be displayed in sound, saving important screen space. If the user is overloaded with visual information then some of this could be displayed in sound as the auditory sense is under-utilised and has spare capacity. One of the advantages of sound is that it is omni-directional, the user does not have to concentrate his/her visual attention on any part of the screen. Sound does not take up any screen space so displaying status information in this way leaves more space for the task the user is performing with the computer.

Hidden information also widens Norman's *Gulf of Evaluation* [34, 35]. Norman defines this thus ([34], p 40):

> "Evaluation requires comparing the interpretation of system state with the original goals and intentions. One problem is to determine what the system state is, a task that can be assisted by appropriate output displays by the system itself."

If information is hidden then users may not be able to determine the state of the system. Providing more information may increase the problem if it is presented in a distracting way. It may make it harder to find the important information needed amongst greater visual clutter. Presenting more and more information on the graphical display and making the display bigger and bigger can result in overload and the user may miss information. One way to overcome this is to use sound. The auditory system can provide the extra capacity needed without increasing the amount displayed visually.

Action Slips

This paper deals with errors that result from hidden information. Information might be hidden by the system, as described above, but it might also be hidden because of errors on the part of the user. Many usability errors with interface widgets occur because of *action slips* [38]. This type of error occurs with expert users who perform many tasks automatically and do not explicitly monitor the feedback from each interaction because the task is well known. Reason ([38], p.8) suggests:

> "Two conditions appear to be necessary for the occurrence of these slips of action: The performance of some largely automatic task in familiar surroundings and a marked degree of attentional capture by something other than the job in hand".

An example of where this type of problem might occur is when scrolling. This is a very common task when interacting with graphical interfaces and so becomes automatic. As mentioned, a necessary condition for the occurrence of a slip is the presence of attentional capture associated with either distraction or preoccupation. In this case the capture would come from preoccupation with the main task being undertaken. This may be, for example, the preparation of a paper and any scrolling is a small, subsidiary part of this. The writer's attention will be focused on the paper being written, not on scrolling. He/she will not monitor all of the feedback from scrolling because the task is well known and attention is focused on the paper being written. This means that scrolling errors can occur.

The discussion in this section has demonstrated some of the problems due to hidden information in the human-computer interface. A method is therefore needed to find out where hidden information exists so that it can be made explicit.

THE STRUCTURED METHOD FOR INTEGRATING SOUND INTO USER INTERFACES

This section describes a structured method that can be used by an interface designer, unskilled in sound design, to create effective sonically-enhanced interfaces. There are two components to the structured method. The first is the analysis technique that identifies any hidden information in an interaction and so where sound should be used. The second is a set of guidelines to help the designer create the sounds required to make the hidden information explicit. The following two sections describe each of these components.

The Event, Status and Mode (ESM) Analysis Technique

Hidden information in an interface can be found by modelling interactions in terms of event, status and mode information. Analysing interactions in this way was first put forward by Dix and colleagues [15, 16, 17]. A brief description of the analysis method follows. Dix, Finlay, Abowd & Beale ([15], p 325) describe it as an 'engineering' level technique. They say: "An engineering approach is built upon theoretical principles, but does not require a deep theoretical background on the part of the designer". Dix's technique is based on formal models of interaction but the interface designer using it does not need to know these in order to employ it effectively. It is also built on what Dix calls 'naïve' psychological knowledge, which is used to predict how particular interface features affect the user. These foundations make the technique powerful and easy to use. A definition of the three components of the technique will now be given.

Events

An event marks something that happens at a discrete point in time. Events are caused by actions on the part of the user (mouse clicks, button presses) or the system (mail arriving). There can be input events (mouse clicks) or output events (a beep indicating an error). The same actions may cause different events under different circumstances. For example, the user clicking the mouse in one window may select an icon but in another may position a cursor.

Dix suggests that there are two time points to an event: The event which occurs and the perception of its occurring. As Dix ([17], p 8) says: "...there may be a lag between the *actual* event and the *perceived* event". If an event occurs and changes the status (for example, new mail arrives putting a message in the mail window) the user might not notice straight away. The actual event is the mail arrival but it is only when he/she looks at the mail window will that the new mail be noticed (the perceived event). In some cases there may be no perceived event, so that actual event is missed by the user. The perceived event may be missed, for example, because the user was not looking at the right place on the screen at the right time. An actual event that does not permanently change the status of the display (for example a screen flash to indicate an error) will be never be perceived by the user if he/she is not looking at the screen at the time it occurs. There may also be a perceived event with no corresponding actual event: The user may have thought he/she heard a beep from his/her computer but it actually was from someone else's. If the event is not perceived by the user then it is hidden and this can result in errors.

Status

Dix describes status information as any phenomenon with a persistent value, for example the location of a mouse, an indicator, a temperature gauge or CPU load monitor. Much of the display in a graphical interface is status information. Status information is what the user can perceive about the state of the system. The state of the system is the complete set of values of the internal variables and components of the system. If there is information about the state that is not displayed as status information then it will be hidden from the user. If important state information is not displayed in the status (hidden information) then errors can occur.

Events may change the status information. An event such as a mouse click in a scrollbar may change the status information displayed in a window because it scrolls. Another example is clicking the 'OK' button in a dialogue box. This causes the box to disappear, changing the status of information displayed on the screen. If an event changes the state of the system then this should be reflected in the status, if it is not then the hidden information could cause errors.

The status (and event) information can be displayed, or *rendered*, in many different ways, for example graphically, textually, sonically or by a combination of these. Earlier in this paper the problems of presenting status information graphically were discussed. Scott & Findlay [40] showed that, if it was rendered graphically, much screen space had to be devoted to make it noticeable. This paper suggests that rendering status information in sound is possible and could overcome some of the problems associated with visual representations.

Modes

It was decided as part of this research that modes should be added to extend Dix's event and status analysis. Modes are very closely linked to events and status (as will be shown below) and they cause some of the main problems of hidden information at the interface. Modes have been investigated ever since Tessler complained about being 'moded-in' in 1981 [43]. However, as Sellen, Kurtenbach & Buxton report [41, 42], even now there is little relevant literature in the area.

A mode is a state within a system in which a certain interpretation is placed on information. For example, in one mode characters typed into an editor may appear on the screen, in another mode the characters may be interpreted as commands. It is often the case that the details of this interpretation are hidden from the user (or perhaps forgotten due to overload of short-term memory, or not noticed because the user was not looking at the required part of the screen). Mode ambiguity occurs when the status does not provide enough information to indicate which mode the system is in. The definition of modes can be extended to cover a set of events and status information. Frohlich (reported in [1], p 38) defines modes as: "States across which different user actions can have the same effect". In one mode a set of user actions is possible and this causes a particular set of events. A set of status outputs describe the state of the system. In another mode the same set of user actions may cause different events and there will be different status feedback. Indeed, a different mode might have a completely different set of user actions. For example, in a wordprocessor, keystrokes might cause characters to be displayed on the screen. The window and the text it displays provide status information about being in wordprocessor mode. If the user switches to a drawing package, the set of legal user actions might change to mouse clicks and drags. Status information would change to reflect the new mode. There would be the graphics displayed in the window, a tool palette and the window itself might be in a different place on the screen. Therefore, a mode groups a set of events and status information.

Johnson [26] and Johnson & Englebeck [27] say that there is no generally agreed definition of mode. The definition given above is useful for the purposes of this paper but is not the only one. One of the most general Johnson describes thus (p 424): "...a system has modes (i.e. is moded) if the effect of a given user-action is not always the same". His definition fits well with the event and status analysis as user actions cause events and these events have different effects depending on the mode. Sellen *et al.* say (p 143):

> "It is not clear that we can ever hope to completely eliminate the problems associated with modes, but it certainly seems possible to reduce them. One obvious solution seems to be to give users more salient feedback on system state".

If the user was given more status feedback then he/she would be less likely to misclassify a situation and the effects of mode ambiguity would be reduced. The consequences of most mode errors at the human-computer interface are only minor inconveniences which are usually easily reversible. However, errors in more complex systems, such as aircraft or nuclear power plants, can have much more serious consequences so their prevention is important. Errors are not the only problem that can occur due to mode ambiguity. As Sellen *et al.* suggest (p 143):

> "In some cases, the user may diagnose the correct mode, but only after experiencing confusion or uncertainty. In such cases, the appropriate measure is in terms of the cognitive effort or decision time required to deduce the system state".

These types of problems are important to consider as solving them would speed up user operation of systems and reduce frustration. It was therefore decided that modes should be included in the event and status analysis so that they could be examined in detail.

Characterisation of the Feedback

Dix's analysis technique finishes when the event and status information has been identified. It shows where errors occur because of incorrect event and status feedback but does not suggest what feedback is needed to correct the problems found. If his technique is to be used as part of a structured method for integrating sound into human-computer interfaces then it must connect to guidelines for sound design. Dix's work is extended here so that when the event, status and mode information has been extracted it is characterised in terms of the feedback needed to present it. The sound guidelines can then be used with this characterisation to design the auditory feedback necessary. The characterisation described here is based around that of Sellen, Kurtenbach & Buxton [41, 42]. The following sections describe the three of Sellen's dimensions we used plus a fourth, added as part of this research. As well as describing the dimensions, they are discussed in terms of the event, status and mode analysis (a more detailed discussion is provided in Brewster [9]).

Action-dependent versus action-independent feedback

Does the feedback depend on a user or system action taking place? Events are action dependent; an action on the part of the user or the system must occur for an event to take place. For example, the user clicks the mouse button and this causes an event such as selecting an icon. Feedback is only given when the event occurs. Status feedback, however, is action-independent. It continues whether there are actions or not. Status may be changed by events but it continues independently of them. For example, an event may cause a dialogue box to be displayed. The status feedback from this will continue until the user presses the OK button regardless of whether the user moves the mouse or presses keys on the keyboard. Modes, like status, are initiated by events and continue until a further event changes the system to a different mode. For sound feedback, action-dependent delivery would mean sound occurred when some action took place. An example is Monk's keying-contingent sound [31]. In an experiment, he associated different sounds to keystroke events to indicate which mode a system was in. Action-independent delivery would mean that feedback was given without an action taking place, for example a constant background tone when in a particular mode.

Transient versus sustained feedback

Is the feedback sustained throughout a particular mode? Events are transient, they occur at momentary, discrete points. Transient delivery is therefore useful for presenting event information. For example, a short beep to indicate an error. Events are atomic: They cannot be interrupted. Status information continues over time so sustained presentation is needed. For example, a window on the screen displaying the contents of a document is sustained. Sustained sounds can be habituated by listeners [14], the user does not actively have to listen to the sound. The sound will be perceived again only when it changes in some way (when an event occurs) or if the user consciously chooses to listen to it. Status information is non-atomic: Actions and events can take place whilst the status information is presented. Modes may be sustained or transient. A mode might last for a long time (like the window) and be sustained or for a short time (a screen button press) and be transient. Modes may be atomic or non-atomic.

Demanding versus avoidable feedback

Can the user avoid perceiving the feedback? Events should be demanding as they mark important occurrences in the system. The classic example of this in the interface is a demanding beep to indicate an error event. The user needs to know that an error has occurred. Status information should be avoidable. It exists over the time and users should be able to choose to sample it only if they want to. For example, the data in a window on the screen should be avoidable. The user can look at the window if required but should not be forced to see it all the time. This is not always the case as the user may not be interested in some events (for example, the arrival of some types of junk email) and may not want to miss some types of status information (for example, an alarm). Modes should be demanding. Often feedback from a mode is avoidable (when it is there at all), the user does not observe that he/she is in the mode and mode errors can then occur.

This aspect of the categorisation can capture the urgency of the information to be presented. The work by Edworthy and colleagues [18, 19] can be used to make sounds demanding (more urgent) or avoidable (less urgent) as required. As discussed above and by Scott *et al.* [39, 40], presenting information visually so that it is demanding can take up much screen space. Presenting demanding information in sound is easier to do as sound is, by its nature, attention grabbing.

Static versus dynamic feedback

Does the feedback change whilst it is being presented? This extra dimension of feedback was added as part of this research because it was not captured by Sellen *et al.*'s classification. Events are static; they only occur for a moment of time and indicate that one particular thing has happened. Status information can be static, for example a window onto a file directory, or dynamic and change over time, for example a CPU load indicator. *Animated icons* [3] are an example of dynamic status feedback. These change to capture the user's attention or to give information about their state over time. One other example is *metawidgets* [23]. In this system, widgets may move to the edges of screen of their own accord if they are not used for a period of time. Modes are static; they do not change their meaning whilst they are presented. Sound can be static or dynamic, for example a constant tone is static and music is dynamic.

The feedback from each of these categories is not necessarily independent. For example, dynamic visual feedback is more demanding than static feedback because the user's eye is drawn to the changing stimulus. In the same way demanding audio feedback captures attention. In order to create demanding feedback a high volume, static sound could be used or, alternatively, a lower volume, dynamic sound. The latter case would be less annoying for other users nearby but just as demanding for the primary user. Therefore, a demanding sound could be created that was both attention-grabbing for the primary user but not annoying for other users at the same time.

This categorisation converts the raw hidden information from the event, status and mode analysis into a structured form which can then be converted into sound. Once the categorisation has been used the designer will be able to create sounds to represent the hidden information. The results from the analysis could be used to improve the graphical feedback from interactions to overcome the problems of hidden information. However, this paper suggests that this would clutter the interface further and overload the visual system.

Earcon Guidelines

Along with the analysis technique to find out where sound should be added there is a second part to the structured method for integrating sound into human computer interfaces. This is a set of guidelines derived from experimental investigations into structured audio messages called *earcons* [11, 12, 13]. Earcons are abstract, synthetic tones that can be used in structured combinations to create sound messages to represent parts of an interface [8]. The main points addressed by the guidelines are:

Timbre: Use musical instrument timbres. Where possible use timbres with multiple harmonics as this helps perception and can avoid masking. Timbres that are subjectively easy to tell apart should be used. For example, on a musical instrument synthesiser use 'brass' and 'organ' rather than 'brass1' and 'brass2'. However, instruments that sound different in real life may not when played on a synthesiser, so care should be taken when choosing timbres. Using multiple timbres per earcon may confer advantages when using compound earcons. Using the same timbres for similar things and different timbres for other things helps with differentiation of sounds when playing in parallel.

Register: If listeners are to make absolute judgements of earcons then pitch/register should not be used. A combination of register and another parameter would give better performance. If register alone must be used then there should be large differences between earcons but even then it might not be the most effective method. Two or three octaves difference give better recognition. Much smaller differences can be used if relative judgements are to be made.

Pitch: Complex intra-earcon pitch structures are effective in differentiating earcons if used along with rhythm or another parameter. The maximum pitch used should be no higher than 5kHz (four octaves above C_3) and no lower than 125Hz-150Hz (the octave of C_4) so that the sounds are not easily masked and are within the hearing range of most of the population.

Take care that the pitches used are possible given the chosen timbre; not all instruments can play all pitches. For example, a violin may not sound good if played at very low frequencies. If a wide range of pitches is needed then timbres such as organs or pianos are effective.

Rhythm and duration: Make rhythms as different as possible. Putting different numbers of notes in each rhythm is very effective. Patterson [36] says that sounds are likely to be confused if the rhythms are similar even if there are large spectral differences. Small note lengths might not be noticed so do not use notes less than 0.0825 sec. However, if the earcon is very simple (one or two notes) then notes as short as 0.03 sec. can be used.

Earcons should be kept as short as possible so that they can keep up with interactions in the interface being sonified. Two earcons can be played in parallel to speed up presentation. Earcons with up to six notes played in one second have been shown to be usable. In order to make each earcon sound like a complete rhythmic unit the first note of each should be accented (played slightly louder) and the last note should be slightly longer [24].

Intensity: Great care must be taken over the use of intensity because it is the main cause of annoyance due to sound. The overall sound level will be under the control of the user (in the form of a volume knob). Earcons should be kept within a narrow range so that if the user changes the volume no sound will be lost and no one earcon will stand out and be annoying.

Listeners are not good at making absolute intensity judgements. Therefore, intensity should not be used on its own for differentiating earcons. If it must be used in this way then there should be large differences between the intensities used. This may lead to annoyance on the part of the user because it contravenes the previous guideline. Some suggested ranges [6] are: Maximum: 20dB above the background threshold and minimum: 10dB above threshold.

One of the main concerns of potential users of auditory interfaces is annoyance due to sound pollution. If intensity is controlled in the ways suggested here then these problems will be greatly reduced [9].

Spatial location: This may be stereo position or full three-dimensions if extra hardware is available. This is very useful for differentiating parallel earcons playing simultaneously. It can also be used with serial earcons, for example each family of earcons might have a different location.

Making earcons attention grabbing: In many cases earcon designers may want their sounds to capture the listener's attention. This can be achieved in different ways. It can be done by using intensity. This is crude but effective (and very common). However, it is potentially annoying for the primary user and others nearby so we recommend other methods. Rhythm or pitch can be used (perhaps combined with lower intensity), for example, because the human auditory system is very good at detecting dynamic stimuli. If a new sound is played, even at a low intensity, it is likely to grab a listener's attention (but not that of a colleague nearby). As another example, if the rhythm of an earcon is changed (perhaps speeding up or slowing down) this will also demand attention.

Other techniques for making sounds attention-grabbing are to use: High pitch, a wide pitch range, rapid onset and offset times for sounds, irregular harmonics and atonal or arrhythmic sounds (for more see [17]). The opposites of most of these can be used to make sounds avoidable but in this case the main parameters are low intensity and regular rhythm.

Compound earcons: When playing serial earcons one after another use a 0.1 second gap between them so that users can tell where one finishes and the other starts. If the above guidelines are followed for each of the earcons that is to be combined then recognition rates will be high. If the above guidelines are followed then earcons played in parallel should also be recognisable.

The output of the ESM analysis technique and the categorisation can be converted into sound using these guidelines. The two components of the structured method for integrating sound into human-computer interfaces have now been demonstrated. As an example, the next section will show the application of the method to sonify a scrollbar.

APPLICATION OF THE STRUCTURED METHOD

Now that the structured method has been described the analysis of a scrollbar will be undertaken to show how it works in practice. This informal analysis technique can be used to investigate known problems with interactions and find what might be causing the mistakes. It can also be used to look at new interactions to find where there are likely to be faults. The technique has been used to analyse many widgets. Analyses will be applicable to many different interfaces because, as Myers' [32] shows, most widgets vary little in their general design across different interfaces.

ESM Analysis

To use the event, status and mode (ESM) analysis technique first think of the 'generic' or perfect widget, for example a button, which provides all the information required and where nothing is hidden. This can be done by creating scenarios depicting the interaction with buttons and mapping out the possible types of events, status and modes that occur. Identify all the event, status and mode information in the interaction and the feedback required to present it using the descriptions given previously. Then do the same for the real button, identifying the information actually present. If there is less in the real button than the ideal, generic one then that is where hidden information exists. If there is more, then there may be redundant information that could possibly be removed. This is similar to the approach taken by Kishi in the SimUI system [29] where a 'model' or expert user's interactions are compared to those of normal users and where the differences occurred are usability problems. One alternative approach to using the method is to first think of the real widget and analyse it in terms of events, status and modes and then categorise the feedback. Then, using this information, construct a generic, ideal, widget that deals with the problems of the real one and makes explicit the hidden information.

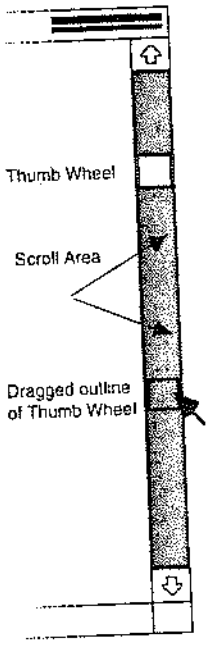

Figure 1: An example scrollbar.

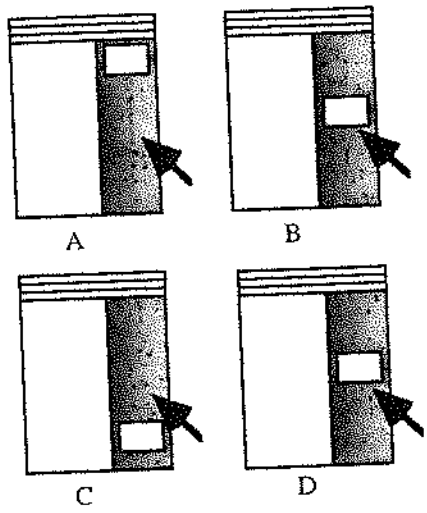

Figure 2: Scrollbar 'kangarooing'.

In the following example, scrollbars are examined using the ESM technique to find out if problems associated with them can be identified and characterised. Scrolling through a document can be achieved in several ways (see Figure 1). The user can click on the arrows at the top or bottom of the scrollbar, in the grey scroll area or drag the thumb-wheel (the white box in the scroll area). The thumb-wheel gives status information about the currently viewed position in the document. There are two common problems associated with scrollbars:

Position awareness in the document: When scrolling through a document it can be hard to maintain a sense of position. The text can scroll by too fast to see (and the thumb only gives an approximate position). Some systems such as Microsoft Word put a page count in the bottom left hand corner of the screen but this is too far from the centre of visual focus so is hidden. One other method, used by MacWrite, is to put the page number in the thumb wheel itself. This is closer to the user's centre of visual focus and therefore should be more effective. The problem with this is that the thumb is very small so only a small amount of data can be displayed. It may also be missed by users if they are looking at the main part of the window and not the scroll bar. It may force users to look at the scrollbar when they really want to look at the screen.

'Kangarooing' with the thumb wheel: Repeatedly clicking in the scroll area above or below the thumb scrolls by a window-sized step. Clicking below the thumb scrolls down in the document and clicking above scrolls up. When the thumb is just above the target position (pointer location) it will scroll down to the window below (because the pointer is below the thumb) and then back up to the window above the target (because the pointer will then be above the thumb) and keep on doing this until the user notices and stops clicking. If the document is scrolling fast then it can be hard to tell this is happening as the user cannot see the text in the window (it is moving too fast to provide any status cues) so the information is hidden. Figure 2 shows an example of kangarooing. In A the user begins to scroll down towards the pointer. In B the thumb wheel is just above the pointer. In C the user has clicked and the thumb has scrolled below the pointer. In D the user clicked again and the thumb scrolled back above the pointer so kangarooing occurred. Unless the one is looking at the thumb it can be hard to recognise that this has happened.

This type of error is an action slip (see above for a description). Expert users do not explicitly monitor the feedback from the scrollbar, they use it automatically. They will click on it an expect to scroll to where they want to be. Experts will not notice when kangarooing occurs because the only way to observe it is to look at the scrollbar. However, their visual attention will be on the document they are working on.

Generic (ideal) scrollbar

Figure 3 shows the ESM analysis of scrollbar kangarooing and loss of position. The mode occurs when the mouse button is clicked in the scroll area. A single click of the mouse scrolls by one window of data. Holding the mouse button down (or clicking the mouse many times) continuously scrolls by multiple windows of data. The mode should

be action-independent (the mode continues until the mouse is moved out of the scroll area), demanding (users should know that they are in the mode), sustained/transient (the mode can be sustained, if the user keeps the mouse button pressed down, or transient if the user just clicks the mouse) and static (the mode will not change until the user moves out of the scroll area).

There are three events. The first is the user clicking in the window scroll area, the second is the thumb reaching the target location and the third the crossing of a page boundary when scrolling. Feedback from the click event should be action-dependent (the user must press the mouse button), demanding (the user actively has to click the mouse), transient (the mouse is only pressed for a short time) and static (the feedback indicating the click does not change). When the thumb reaches the target (the cursor position) a demanding event should be given to alert the user. This event feedback should also be action-dependent (the thumb reaches the cursor position because the user clicks), transient (the event lasts a short time) and static (the feedback does not change). The event of crossing a page boundary occurs when the user scrolls passed a page boundary. The feedback should be action-dependent (the user must click the mouse), demanding (the user should know when a page boundary has been crossed), transient (crossing the boundary lasts a short time) and static (the feedback just indicates that a boundary has been crossed).

There are three types of status information: Information from the thumb and when it moves, information from what is in the window and when that changes, and the position in document indication. This may be, for example, a page count in the thumb or elsewhere in the window. These three types of status feedback are changed by the user clicking in the scroll area. This event causes a change in the status of the window and scrollbar thumb. It may change the document position indicator if a page boundary is crossed. Thumb wheel feedback should be action-independent (it gives information about position in document no matter what the user is doing), demanding (because the thumb is the primary means of interaction with the scrollbar), sustained (feedback from the thumb lasts for as long as the scrollbar is displayed) and dynamic (the feedback can change because of the click event). The feedback from the thumb should be demanding because it is the main source of feedback from the scrollbar giving information about where the user is in the document.

Status feedback from the window is similar. It is action-independent (it gives feedback no matter what the user is doing, although this changes when Event 1 of Figure 3 occurs), demanding (it takes up a large area of the screen), sustained (the window lasts for a period of time) and dynamic (Event 1 will cause the status information to change). The status feedback from the window is classed as demanding because it takes up a large area of the screen. The user can, of course, not look at the screen but if he/she is looking at it then the window is likely to be the area of focus. Event 1 will cause the window to scroll which is a very demanding occurrence because a large area of the screen changes.

Generic (ideal) scrollbar: **Mode** When mouse button clicked in scroll area **Event** 1. Click in scroll area 2. Thumb reaches target 3. Cross page boundary **Status** 4. Scrollbar thumb 5. Window 6. Position in document indication	**Feedback:** **Mode** Action-independent, demanding, sustained/transient, static **Event** Action-dependent, demanding, transient, static for all events **Status** 4. Action-independent, demanding, sustained, dynamic 5. Action-independent, demanding, sustained, dynamic 6. Action-independent, avoidable, sustained, dynamic
Real scrollbar: **Mode** When mouse button clicked in scroll area **Event** 1. Click in scroll area **2. No thumb reaches target event** 3. Cross page boundary **Status** 4. Scrollbar thumb 5. Window 6. Position in document indication	**Feedback:** **Mode** Action-independent, **avoidable**, sustained, static **Event** 1. Action-dependent, demanding, transient, static 3. Action-dependent, **avoidable**, transient, static **Status** 4. Action-independent, **avoidable**, sustained, dynamic 5. Action-independent, demanding, sustained, dynamic 6. Action-independent, avoidable, sustained, dynamic

Figure 3: ESM analysis of scroll bar loss of sense of position and 'kangarooing'. The emboldened and underlined items show the differences between the generic and real interactions.

The document position indicator should be action-independent (the feedback is given no matter what the user does), avoidable (the user should be able to sample the status information to find out what the current page is), sustained (the feedback lasts for as long as the window is displayed) and dynamic (the feedback may change in response to event 3). The page indicator changes its feedback in response to the crossing of a page boundary. The status feedback should be avoidable: Users will not want the information forced on them, they will want to be able to sample it if

necessary. The page boundary event should, however, be demanding because the user should know when a new page has been reached so that they do not lose their sense of position in the document.

Real scrollbar

Figure 3 shows the ESM analysis of the real interaction. The emboldened and underlined items show the differences between the real and generic interactions. In the real scrollbar the mode is the same as above (i.e. when the mouse button is clicked in the scroll area). However, this time the feedback on the mode is avoidable and this means that kangarooing and loss of position can occur. There is no event to indicate that the thumb has reached the target, which is why kangarooing can occur. This event information is hidden: The user will not know that the thumb has moved to the target position (usually the pointer location in the scroll area) unless he/she happens to be looking at it. The event feedback indicating when a page boundary has been crossed is also avoidable. The user will not see the document position indicator change because it is outside the area of visual focus. Often a dotted line is displayed in the window on the screen to show a page boundary. If the user is scrolling rapidly through the file then this too is likely to be missed.

The status information is the same as before but feedback from the thumb is avoidable: It is easy to avoid seeing the thumb move because it is small. The same is true of the document position indicator (it is again avoidable) as it is out of the area of visual focus. The status feedback from the window is very demanding and when it changes (due to the click event) the change captures the user's attention. Kangarooing occurs because the event indicating when the thumb reaches the target location does not exist in the real scrollbar. This is made worse because status feedback from the thumb is avoidable. If this was demanding the user would see when the thumb reached the target so kangarooing could be avoided. Loss of sense of position occurs because the page indicator is outside the area of focus and the document can scroll by very quickly. The very demanding nature of the scrolling window causes the user not to look at the page count.

Sound could be used to overcome the problems associated with kangarooing. If a demanding sound was added to indicate the event of the thumb reaching the target location then the user would hear it and stop clicking the mouse. For example, if a low-pitched sound was played when scrolling down through a document and a high-pitched one when scrolling up then kangarooing would be perceived as an out-of-sequence high or low tone.

The crossing of a page boundary event could be given a demanding sound so that users know when a new page has been reached. This would help them keep a sense of position in the document. The document position indicator could itself be given an avoidable, action-independent sound. The user could then perceive the page without taking his/her visual attention away from the primary task. This could be done by having a sustained, dynamic tone, the pitch of which could change as a page boundary was crossed. This feedback would not be annoying, even though it would be sustained, because it would not need to be demanding: It could be made to fade into the background of consciousness. The user would only perceive it when he/she wanted to listen for the page sound or when a page boundary was crossed. It would also indirectly give information about the thumb and its position. Using this method it would be easy for users to move, say, two pages through the document; they would just listen for two page boundary sounds.

Earcon Creation

Now that the ESM analysis of the scrollbar has been carried out, earcons must be created to overcome the problems identified. Two types of sounds were needed to solve the problems described above: One to give information about when the thumb reached the target location (to avoid kangarooing) and one to give page scrolling and document position indication. An electric organ timbre was chosen for the sounds. This has a wide range of possible pitches and is easily recognisable.

The first sound was a fixed tone of duration 9/60 sec. and was used to indicate a window scroll event. The sound was kept short as it indicated a simple event. This would also mean that it could keep up with the interactions taking place. If a user scrolled towards the bottom of a document a low-pitched note, C_4 (130Hz), was played. When scrolling up a high-pitched note C_0 (2093Hz) was played. These sounds were within the range specified in the guidelines and far enough apart to easily be differentiated. High pitch was used as up and low pitch as down because of the natural tendency to perceive higher pitch as higher spatial location [30]. The intensity of the sounds was raised over the background, continuous document position sound. This, combined with the change in pitch, made the sound demanding. Using the combination of pitch and intensity meant that the earcons did not have to be as loud as if only intensity had been used. They would then be less annoying for other users nearby. If a user was scrolling downwards towards a target location he/she would hear the repeated low-pitched sound. If kangarooing occurred then the user would hear a demanding high-pitched tone when not expected.

A low intensity continuous tone was used to give status information about the current page. To indicate a page boundary event the background tone was increased in volume for two beeps of 9/60 sec. each to demand the

listener's attention. It then decreased again to just above threshold level so that it could be habituated. The different number of notes differentiated this earcon from the previous window scroll sound. Again the sound was short so that it did not hold up the interaction. The notes played when scrolling towards the bottom of the document decreased in pitch from B_1 (1975Hz) to C_4 (130Hz) when a page boundary was crossed. The notes played cycled through the scale of C major. So, for example, when scrolling down from the top of the document, the first note played would be B_1 (1975Hz), then A_1, G_1, F_1, E_1, D_1 and on to C_2 of the octave below. The reverse occurred when scrolling up from the bottom of the document. When the scrollbar was clicked the thumb sound was played first followed by the page boundary sound after a 9/60 sec. delay (if a page boundary had been crossed).

AN EXPERIMENT TO EVALUATE THE STRUCTURED METHOD

The structured method has made predictions about where, and what, sound should be used in a scrollbar. We need to test such a scrollbar to see if these predictions are correct and if the sounds improve usability.

Participants

Twelve participants were used. They were postgraduate students from the Department of Computer Science at the University of York. All had more than three years experience of graphical interfaces and scrollbars. Expert participants were used because the type of error studied here is an action slip (see above).

Participants	Condition 1	Condition 2
Six Participants	Auditory scrollbar Train & Test	Visual scrollbar Train & Test
Six Participants	Visual scrollbar Train & Test	Auditory scrollbar Train & Test

Figure 4: Format of the experiment.

Tasks

Participants were given two types of task. The first, which will be called the *Search Tasks*, involved the participants visually searching through a file of data to find significant features. These features were such things as whole line of 'a's together. When the target was found the participants had to say which page the target occurred on. The other tasks, which will be called the *Navigate Tasks*, involved participants being given instructions to go to a specific point on a specific page and read the data that was there. For example, participants were asked to go to page seven and read the first six characters of the first line. Along with these absolute navigation tasks relative tasks were also given. For example, participants were asked to go up four pages from the current page and read the first six characters of the last line. These two types of tasks cover some of the main ways users interact with scrollbars. They might be searching through a document to find something or they might be looking for a specific page to find the data they want. The data were described to the participants as 'experimental results data'. The rationale given to the participants for the tasks was that they were searching through the data to find significant features for analysis.

Experimental Design and Procedure

The experiment was a within-subjects repeated-measures design (see Figure 4). A simple document browser was created on an Apple Macintosh, based around TinyEdit, an example program supplied by Symantec Corporation with the Think Pascal compiler (see Figure 5). This browser allowed participants to navigate around a document using a scrollbar and indicated page boundaries with a dotted line, in a similar way to many wordprocessors. The scrollbar used in the browser only allowed clicking in the grey scroll area above or below the thumb wheel to scroll by a window of data either way. The participants could not drag the thumb wheel or scroll by lines using the arrows. This was done because the experiment was to investigate kangaroo problems and these only occur when clicking in the scroll area.

The data files used in the browser were made up of groups of three lines of 30 randomly generated 'a' to 'f' characters separated by a blank line. The test files had twelve pages of data where pages were 50 lines long and windows 33 lines long. Therefore, scrolling by a window did not necessarily mean that a new page boundary would be reached each time. The data was displayed in 12 point Geneva font. There was a different test file for training, the auditory condition and the visual condition so that participants would not be able to learn their way around the data files.

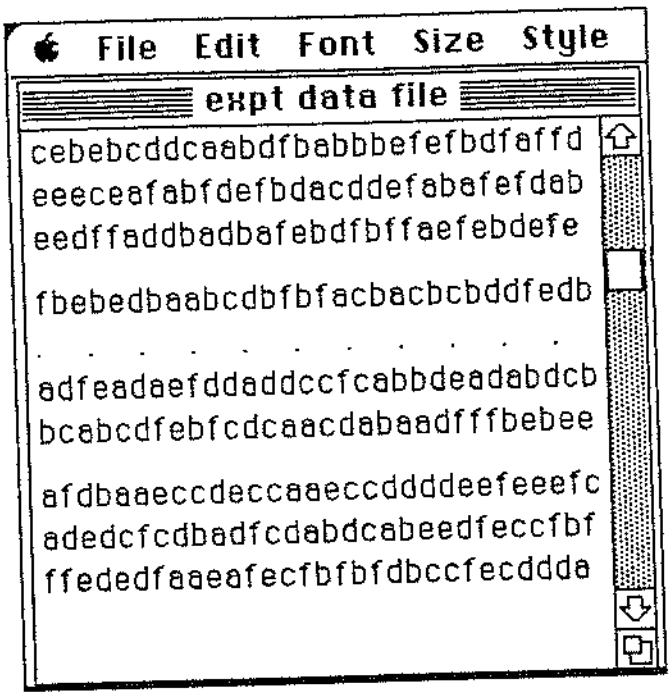

Figure 5: The browser program (reduced in size). It shows example data and a page boundary marked by dots.

Measures

In order to get a full range of quantitative and qualitative results time, error rates and workload measures [25]were used. Time and error rate reductions would show quantitative improvements and workload reductions would show qualitative improvements [5].The total time taken by each participant was recorded by the system (as described below). Two types of errors data were collected: The number of times kangarooing occurred (kangaroo errors) and the number of times the wrong page was chosen (wrong page errors).

The NASA Human Performance Research group [33] break workload down into six different factors: Mental demand, physical demand, time pressure, effort expended, performance level achieved and frustration experienced. NASA have developed a measurement tool, the NASA-Task Load Index (TLX) for estimating these subjective factors. We used this but added a seventh factor: Annoyance. One of the main concerns of potential users of auditory interfaces is annoyance due to sound pollution. This is often given as a reason for not using sound at the human-computer interface [28]. In the experiment described here the annoyance due to auditory feedback was measured to find out if it was indeed a problem. In addition to these seven factors we also asked our participants to indicate, overall, which of the two interfaces they felt made the task easiest. Participants had to fill-in workload charts after both conditions of the experiment.

Visual condition

In the visual condition, participants used an ordinary graphical Macintosh scrollbar (but restricted as described above). Training was given in both types of task before the main test was started. The experimental procedure was described and then sample Search and Navigate tasks were undertaken using a training data file. In the main test participants were given a task by the experimenter and when they were ready to start they pressed ⌘Y to start a timer. When they had completed the task they pressed ⌘Y again, the timer was turned off and the time recorded by the system. Errors were recorded by the experimenter. A participant was given the search task questions first and then the navigate ones. When the participant found a target in the search task he/she gave the page number to the experimenter. If it was incorrect then the correct page number was given by the experimenter. This was necessary so that the participant started from the correct page when searching for the next target. For the navigate tasks, the participant gave the required six characters. If these were incorrect then the participant had to search until he/she found the correct page.

Auditory condition

The sonically-enhanced scrollbar described above was used. In the initial training of participants for this condition the feedback provided by the scrollbar was described in detail. The training and testing then proceeded as described above for the visual condition. All the sounds used were played on a Roland D110 multi-timbral sound synthesiser.

The sounds were controlled by an Apple Macintosh via MIDI through a Yamaha DMP 11 digital mixer and presented to participants by loudspeakers.

Experimental Hypotheses

If the errors identified by the structured method were real problems then fixing them would improve usability. The method suggested that recovery from kangaroo errors should be quicker because users receive demanding feedback indicating when the errors happen rather than noticing later on when they were not where they expected to be. Participants can therefore correct the errors more quickly. The number of kangaroo errors should be unchanged.

Participants should better be able to maintain their sense of position in the document with more page feedback and therefore give fewer wrong page answers. If participants lost their sense of position the time cost was high. For example, they would have to go back to the top of the data file and work out their position from there. This would take much time. Therefore, if they did not lose their sense of position, time to complete tasks should be reduced. The demanding audio feedback should make it easier for participants to perceive page boundaries and so make fewer wrong page errors.

The workload felt by participants should be reduced as the extra feedback provided information that they needed. Participants would have to expend less effort recovering from errors and remembering whereabouts in the document they were. Physical demand and time pressure would be unaffected as they were unchanged across conditions. There would be no increased frustration or annoyance due to the addition of sound as the auditory feedback provided information that the participants needed.

Results

TLX results

Figure 6 shows the average score for each of the workload categories plus the extra ones we added. They were scored in the range 0-20. Paired T-tests were carried out on the auditory versus visual conditions for each of the categories. Mental demand was assigned the highest mark of all the workload scores indicating that the experimental task itself was difficult. It showed a significant decrease in the auditory condition over the visual ($T(11)=3.23$, $p=0.008$). Nine of the twelve participants rated the auditory condition lower in effort than the visual but this failed to reach significance ($T(11)=1.83$, $p=0.09$). There were no significant differences in any of the other workload categories.

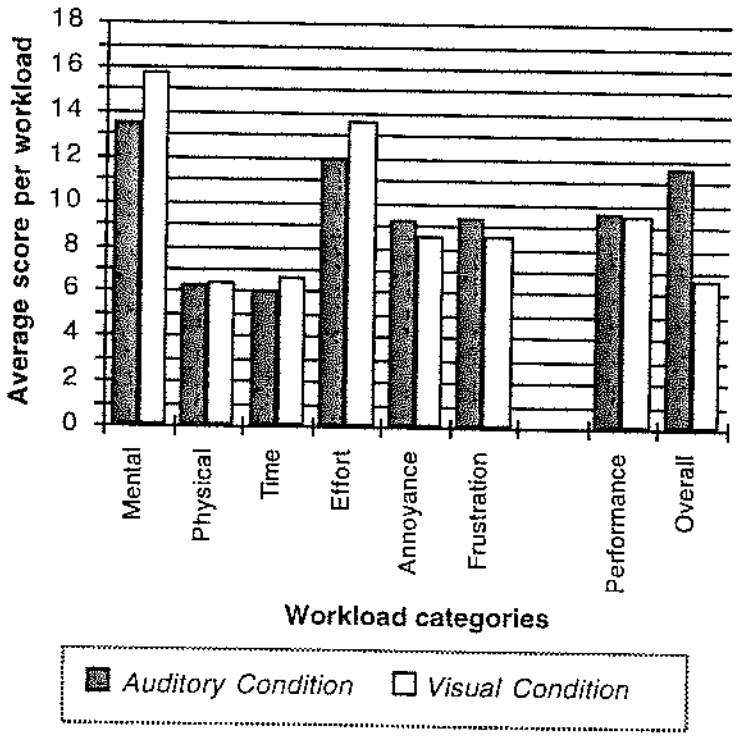

Figure 6: Average TLX workload scores for the auditory and visual conditions of the experiment. In the first six categories higher scores mean higher workload. The final two categories, performance and overall, are separated because higher scores mean less workload.

The annoyance for the auditory condition was not significantly different to the visual condition ($T(11)=0.516$, $p=0.615$). Five participants rated the auditory condition more annoying than the visual and three rated the visual more annoying than the auditory. There was a difference in terms of overall preference. In this case, the participants were asked to rate which scrollbar made the task the easier. Here the auditory scrollbar was significantly better than the visual one ($T(11)=2.55$, $p=0.02$).

Timing and error results

Along with workload tests, conventional measures of time and error rates were taken. Two kinds of errors were recorded: Kangaroo errors and wrong-page errors (see Figure 7). Figure 8 shows the total times taken by each of the participants in the two conditions for the search tasks. Nine of the twelve participants performed faster in the auditory condition but there was no significant difference in time scores at the 95% level ($T(11)=1.846$, $p=0.09$). However, an F-test between the auditory and visual conditions across participants showed a significant reduction in the variance in the auditory condition ($F(11,11)=3.98$, $p=0.05$).

To find out if any underlying differences were hidden in the overall timing data a more detailed analysis was undertaken. The average time taken to answer a question where errors occurred was calculated for each question in both conditions of the search tasks. Both types of errors were included in this analysis because of the small numbers of kangaroo errors. There were no significant differences between the conditions in time taken to answer questions with errors ($T(2)=1.24$, $p=0.33$) or to answer questions where there were no errors ($T(2)=1.39$, $p=0.29$).

Figure 8 also shows the total times for the two conditions in the navigate tasks. In these tasks there was a significant difference between the times taken. A paired T-test showed the auditory condition was significantly faster than the visual ($T(11)=2.29$, $p=0.04$). As before, there was also a significant reduction in the variance in the auditory condition ($F(11,11)=5.43$, $p=0.05$). To find whether the decrease in time taken for the auditory condition was due to faster recovery from errors, a more detailed analysis was undertaken. Recovery from errors was significantly faster in the auditory than in the visual condition ($T(9)=2.61$, $p=0.02$). The average time taken to answer questions with no errors was also calculated. A paired T-test showed that the auditory condition was again significantly faster than the visual ($T(9)=4.18$, $p=0.002$).

In the search tasks there was no reduction in the number of wrong page errors between conditions (see Figure 7). In the navigate tasks there was again no significant differences in the error rate between the two conditions but there was a reduction. The number fell from 51 to 40 in the auditory condition but this failed to reach significance.

Tasks/ Conditions	Search		Navigate	
	Wrong page	Kangaroos	Wrong page	Kangaroos
Auditory	13	5	40	4
Visual	11	3	51	8

Figure 7: Totals of wrong page and kangaroo errors in both conditions of the experiment.

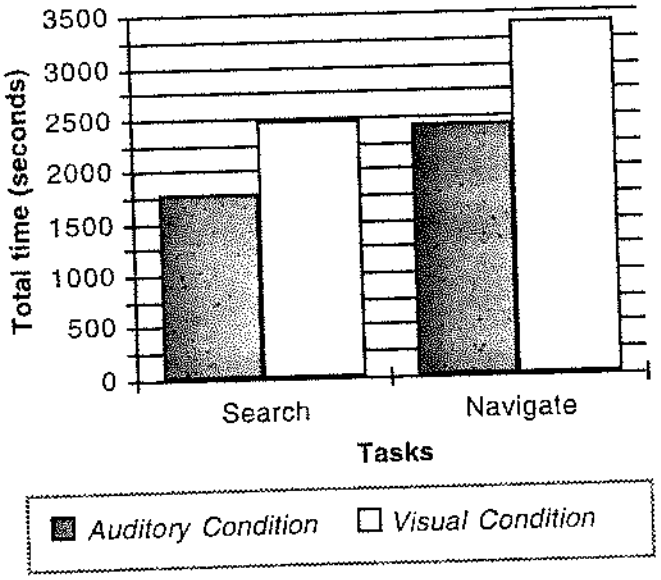

Figure 8: Total times for the search and navigate tasks in the experiment.

Discussion

The workload results indicated that the auditory scrollbar reduced the workload of the task. Mental demand (which dealt with how much mental and perceptual activity was required) was significantly reduced. This could be due to it being easier for participants to hear page boundaries than it was to see them as the feedback was more demanding. Participants also got more feedback about kangaroo errors so making it less effort to recover from them. This confirmed the hypothesis that extra auditory feedback would lower workload. Although participants felt their performance was no better in the auditory condition than in the visual, they had an overall preference for the auditory scrollbar because it lowered mental demand and there was some decrease in effort expended. These factors indicated that an auditory enhanced scrollbar would be an effective addition to an interface and could lower the workload therefore freeing-up cognitive resources for other tasks.

There was no significant difference in the annoyance or frustration felt by participants in the auditory condition. This indicated that auditory feedback, and especially constant auditory feedback, was not necessarily annoying when used at the interface. This confirmed the hypothesis that auditory feedback would not be annoying if it provided useful information to the user.

The significant reduction in time for the auditory condition in the navigate tasks indicated that the sonically-enhanced scrollbar improved performance. This is again evidence to suggest that auditory scrollbars are an effective extension to standard visual ones. The times for the search tasks were not significantly different. This may have been due to the nature of the task. A participant was required to visually search through the data file to find a target. The advantage conferred by sound may have been lost in the overall time to do the visual searching as this took up a large proportion of the time for this task. Position awareness within the document was bound up in this. The advantages due to sound were small and therefore lost in the large times for visual searching. In the navigate tasks, where the participants had to find a specific page, searching was based on page boundaries so there was a better comparison between the auditory and visual conditions.

There were no significant differences between conditions in the time taken to recover from errors in the search tasks. As described previously, the time to do the searching might have been the problem here. In the navigate tasks the auditory group was significantly faster overall. When this result was investigated in more detail the auditory group was found to be significantly faster at recovering from errors than the visual. The auditory group also performed better when there were no errors. It seems that the sounds helped increase general performance with the scrollbar.

One problem with the error analysis was that the frequency of kangaroo errors was too low to be a good measure. For example, in the search tasks there was fewer than one error per participant in each of the conditions. It turned out to be very difficult to generate many kangaroo type errors. It could be that, as the participants were experienced scrollbar users, they had developed strategies for avoiding kangarooing in their everyday work which they used in the experiment. However, two participants did say that the sounds helped them identify when a kangaroo error had taken place. These problems generating kangaroo errors meant that it was difficult to test the hypothesis that recovery from such errors would be quicker. They had to be combined with wrong page errors and an overall analysis performed.

There were no differences between the conditions in the number of wrong-page errors. It may have been that participants counted the page boundaries whether they saw them or heard them, but it just took longer when they had to do it visually. This may have been one of the reasons for improved performance in the navigate tasks for the auditory condition. Further investigation of errors is therefore necessary.

It is noteworthy that there were significant differences between the auditory and visual conditions in terms of variance on both tasks. Eight of the twelve participants showed less variability in the auditory condition. However, a Sign test between conditions across participants failed to reach significance. There is an indication that the variability has been reduced and further experiments would be needed to investigate this.

Justification of the Structured Method

The sonically-enhanced scrollbar was designed to overcome the problems identified by the structured method. Do the experimental results show that the method works? The addition of sound produced a significant improvement in performance in one of the tasks and a decrease in the overall variability in both tasks. The mental workload required to perform the tasks was significantly less when sound was used and overall preference was for the auditory scrollbar. The results indicate that the addition of sound was successful. The structured method identified actual problems due to hidden information and then provided for the creation of effective earcons to fix them. One area which needs further investigation is error rates. The method predicted that the number of wrong-page errors should be lower but the results failed to demonstrate this because insufficient errors were generated in the experiment.

The structured method has also been used to identify and correct some of the problems with graphical buttons [10]. An experiment to investigate the effectiveness of sonically-enhanced buttons showed that error recovery was again faster in the auditory condition and users also had a strong preference for the sonically-enhanced buttons. The results from this give more proof that the structured method is effective at finding errors and suggesting improvements.

FUTURE WORK

In addition to giving information about page boundaries other events could be indicated when scrolling. In a programming language editor, for example, events such as when a new procedure or function was reached could be displayed in sound. The sounds could also be added to dragging in the scrollbar. If the user dragged the thumb wheel over a page boundary then the sound could change to indicate this. Currently, the scrollbar only allows sounds for 21 pages due to its use of pitch. This could be extended by using different rhythms or intensities along with pitch so that bigger documents could be dealt with. The continuous tone of the page indicator could be made to fade into the background of consciousness by lowering its volume if the user stayed on the same page for a period of time. Currently the sound remains at the same volume and lowering it would help habituation.

CONCLUSIONS

This paper described a method to allow interface designers to add sounds to their interfaces. Sound no longer has to be added in an *ad hoc* way by individual designers. It can be added in an effective way by following the two parts of the structured method. The designer can analyse an interaction in terms of event, status and mode information, categorise the information and from this use the earcon guidelines to create effective sounds. This method for adding sound was applied to a scrollbar and some of the associated problems were corrected. To make sure that the method for adding sound was effective an evaluation of a sonically-enhanced scrollbar was conducted.

A sonically-enhanced scrollbar was tested and found to significantly reduce the overall time taken and the time taken to recover from errors on certain tasks. It also significantly reduced the mental workload and was rated with a significantly higher preference score than a standard visual scrollbar. There was also no increased annoyance due to sound. This indicated that the integration of auditory feedback into graphical widgets was likely to provide more usable interfaces. The results from the experiment suggest that the structured method is effective in identifying areas in the interface where problems occur. Until now there was no structured approach to adding sound to an interface, it was done in an *ad hoc* way by individual designers. The results of the work described here show that widgets that combine both auditory and visual feedback are more effective because they make use of the natural way that humans deal with information in everyday life.

ACKNOWLEDGEMENT

This work was supported by SERC studentship 90310837.

REFERENCES

1. Alty, J. (1991). Multimedia-What is it and how do we exploit it? In D. Diaper & N. Hammond (Eds.), *Proceedings of HCI'91*, Edinburgh: Cambridge University Press, pp. 31-44.

2. Alty, J.L. & McCartney, C.D.C. (1991). Design of a multi-media presentation system for a process control environment. In *Eurographics multimedia workshop, Session 8: Systems*, Stockholm.

3. Baecker, R., Small, I. & Mander, R. (1991). Bringing icons to life. In *Proceedings of CHI'91*, New Orleans: ACM Press, Addison-Wesley, pp. 1-6.

4. Barfield, W., Rosenberg, C. & Levasseur, G. (1991). The use of icons, earcons and commands in the design of an online hierarchical menu. *IEEE Transactions on Professional Communication*, 34(2), pp. 101-108.

5. Bevan, N. & Macleod, M. (1994). Usability measurement in context. *International Journal of Man-Machine Studies*, 13(1 & 2), pp. 123-145.

6. Blattner, M. & Dannenberg, R.B. (1992). Introduction: The trend toward multimedia interfaces. In M. Blattner & R. B. Dannenberg (Eds.). *Multimedia Interface Design*, pp. xvii-xxv. New York: ACM Press, Addison-Wesley.

7. Blattner, M., Papp, A. & Glinert, E. (1992). Sonic enhancements of two-dimensional graphic displays. In G. Kramer (Ed.), *Auditory Display, sonification, audification and auditory interfaces. The Proceedings of the First International Conference on Auditory Display*, Santa Fé Institute, Santa Fé: Addison-Wesley, pp. 447-470.

8. Blattner, M., Sumikawa, D. & Greenberg, R. (1989). Earcons and icons: Their structure and common design principles. *Human Computer Interaction*, 4(1), pp. 11-44.

9. Brewster, S.A. (1994) *Providing a structured method for integrating non-speech audio into human-computer interfaces*. PhD Thesis, University of York.

10. Brewster, S.A., Wright, P.C., Dix, A.J. & Edwards, A.D.N. (1994). The sonic enhancement of graphical buttons. In *Accepted for publication at Interact'95*, Lillehammer, Norway.

11. Brewster, S.A., Wright, P.C. & Edwards, A.D.N. (1992). A detailed investigation into the effectiveness of earcons. In G. Kramer (Ed.), *Auditory display, sonification, audification and auditory interfaces. The Proceedings of the First International Conference on Auditory Display*, Santa Fé Institute, Santa Fé: Addison-Wesley, pp. 471-498.

12. Brewster, S.A., Wright, P.C. & Edwards, A.D.N. (1993). An evaluation of earcons for use in auditory human-computer interfaces. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel. & T. White (Eds.), *Proceedings of InterCHI'93*, Amsterdam: ACM Press, Addison-Wesley, pp. 222-227.

13. Brewster, S.A., Wright, P.C. & Edwards, A.D.N. (1993). Parallel earcons: Reducing the length of audio messages. *Submitted to the International Journal of Man-Machine Studies.*

14. Buxton, W., Gaver, W. & Bly, S. (1991). Tutorial number 8: The use of non-speech audio at the interface. In *Proceedings of CHI'91*, New Orleans: ACM Press: Addison-Wesley.

15. Dix, A., Finlay, J., Abowd, G. & Beale, R. (1993). Chapter 9.4 Status/Event Analysis. In *Human-Computer Interaction*, pp. 325-334. London: Prentice-Hall.

16. Dix, A.J. (1991). Chapter 10: Events and Status. In *Formal Methods for Interactive Systems*, pp. 239-270. London: Academic Press.

17. Dix, A.J. (1992). Beyond the Interface. In *Proceedings of IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, 10-14 August 1992*, Ellivuori, Finland.

18. Edworthy, J., Loxley, S. & Dennis, I. (1991). Improving auditory warning design: Relationships between warning sound parameters and perceived urgency. *Human Factors*, 33(2), pp. 205-231.

19. Edworthy, J., Loxley, S., Geelhoed, E. & Dennis, I. (1989). The perceived urgency of auditory warnings. *Proceedings of the Institute of Acoustics*, 11(5), pp. 73-80.

20. Gaver, W. (1989). The SonicFinder: An interface that uses auditory icons. *Human Computer Interaction*, 4(1), pp. 67-94.

21. Gaver, W. (1992). Using and creating auditory icons. In G. Kramer (Ed.), *Auditory Display, sonification, audification and auditory interfaces. The Proceedings of the First International Conference on Auditory Display*, Santa Fé Institute, Santa Fé: Addison-Wesley, pp. 417-446.

22. Gaver, W., Smith, R. & O'Shea, T. (1991). Effective sounds in complex systems: The ARKola simulation. In S. Robertson, G. Olson, & J. Olson (Eds.), *Proceedings of CHI'91*, New Orleans: ACM Press, Addison-Wesley, pp. 85-90.

23. Glinert, E. & Blattner, M. (1992). Programming the multimodal interface. In *ACM MultiMedia'93*: ACM Press, Addison-Wesley, pp. 189-197.

24. Handel, S. (1989). *Listening: An introduction to the perception of auditory events*. Cambridge, Massachusetts: MIT Press.

25. Hart, S.G. & Wickens, C. (1990). Workload assessment and prediction. In H. R. Booher (Eds.), *MANPRINT, an approach to systems integration*, pp. 257-296. New York: Van Nostrand Reinhold.

26. Johnson, J. (1990). Modes in Non-Computer Devices. *International Journal of Man-Machine Studies*, 32(4), pp. 423-438.

27. Johnson, J. & Engelbeck, G. (1989). Modes Survey Results. *ACM SIGCHI Bulletin*. 20(4), pp. 38-50.

28. Jones, D. (1989). The Sonic Interface. In M. Smith & G. Salvendy (Eds.), *Work with computers: Organizational, Management, Stress and health aspects*,. Amsterdam: Elsevier Science publishers.

29. Kishi, N. (1992). SimUI: Graphical user interface evaluation using playback. In *Proceedings of the Sixteenth Annual International Computer Software & Applications Conference*, Chicago, Illinois: IEEE Computer Society, pp. 121-127.

30. Mansur, D.L., Blattner, M. & Joy, K. (1985). Sound-Graphs: A numerical data analysis method for the blind. *Journal of Medical Systems*, 9, pp. 163-174.

31. Monk, A. (1986). Mode Errors: A user-centered analysis and some preventative measures using keying-contingent sound. *International Journal of Man-Machine Studies*, 24, pp. 313-327.

32. Myers, B. (1990). All the widgets. *ACM SIGRAPH Video Review*, CHI'90 Special Issue(57).

33. NASA Human Performance Research Group (1987). *Task Load Index (NASA-TLX) v1.0 computerised version*. NASA Ames Research Centre.

34. Norman, D.A. (1986). Chapter 3: Cognitive Engineering. In D. A. Norman & S. W. Draper (Eds.), *User-centered system design*, pp. 31-61. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

35. Norman, D.A. (1988). *The psychology of everyday things*. USA: Basic Books.

36. Patterson, R.D. (1982). *Guidelines for auditory warning systems on civil aircraft* (CAA Paper No. 82017). Civil Aviation Authority, London.

37. Portigal, S. (1994) *Auralization of document structure*. MSc. Thesis, The University of Guelph, Canada.

38. Reason, J. (1990). *Human Error*. Cambridge, UK: Cambridge University Press.

39. Scott, D. (1993). Status conspicuity, peripheral vision and text editing. *Behaviour and Information Technology*, 12(1), pp. 23-31.

40. Scott, D. & Findlay, J.M. (1991). Optimum display arrangements for presenting status information. *International Journal of Man-Machine Studies*, 35, pp. 399-407.

41. Sellen, A., Kurtenbach, G. & Buxton, W. (1992). The prevention of mode errors through sensory feedback. *Human Computer Interaction*, 7, pp. 141-164.

42. Sellen, A.J., Kurtenbach, G.P. & Buxton, W. (1990). The role of visual and kinesthetic feedback in the prevention of mode errors. In D. Diaper, D. Gilmore, G. Cockton, & B. Shackel (Eds.), *Human Computer Interaction: Interact'90*, Cambridge, UK: Elsevier Science Publishers B.V. (North Holland), pp. 667-673.

43. Tessler, L. (1981). The SmallTalk environment. *Byte* (August), pp. 90-147.

44. Thimbleby, H. (1990). *User Interface Design*. New York: ACM Press, Addison-Wesley.