Easy for everyone: using components to offer specialised interfaces for software.

Peter Bagnall, Guy Dewsbury, Ian Sommerville

Computing Department, Lancaster University

Abstract

Traditional technology has tended to be developed from the supply side. Technology companies have developed applications that possess a functionality which is then marketed to the wider population. Unfortunately, this technology tends to be designed for a standardised user and systems that can be used by a wider group of people tend to be built as one-off systems or as special needs cases. This paper explores the lack of heterogeneity in the design aspects of user interfaces with reference to communication systems and suggests a more inclusive approach to designing EAT communication system, based on our work within the DIRC[1] project.

Introduction

One problem with universal design is the necessity to compromise between the needs of able bodied and those with some impairment. This makes perfect sense in public areas, where many people, of varying ability, will be interacting with a device. But in the privacy of one's home, solutions tailored to the individual are more appropriate. The telephone, for example, has tended to be reduced in size, with smaller keys and smaller displays. Mobile phones are a really useful commodity, allowing people to communicate from any location, unless they have poor eyesight, poor, dexterity or poor hearing[2]. Electronic interfaces have, in principle, unprecedented abilities to adapt their interfaces to meet the needs of individual users. However, these abilities are barely being exploited in commercial systems, largely due to the cost implications of creating a large range of different interfaces. Using component architectures should however enable more cost-effective deployment of unique systems which can still work together. The aim of this paper is to present a possible approach, not to present a completed work.

Encouraging Heterogeneity

Current software user interfaces are largely homogeneous. All Windows users have essentially the same desktop interface. Even across versions there are few significant differences. While there are minor adaptations in both systems to help vision impaired users, and extra hardware and software can be added to provide alternative input and output methods the ability to significantly modify the desktop system has not been exploited to help disabled users. That it is possible is proven by some of the extensions Microsoft themselves have produced, such as the so called *powertoys*[3], and by a number of third party desktop replacements such as *LiteStep*[4].

A more basic problem is that most software only works with a single medium. Taking word processors as an example, they are designed foremost for creating paper documents, and as such support the concept of pages, and all the various layout options that are available on paper. It is telling that most academic journals are strict about the precise format in which submissions are made. Using a word processor to create documents, even for a medium such as the web which

shares at least some properties with paper, is rather clumsy, since word processors typically lack features to manage websites, a facet that is fundamental to the medium.

While much progress has been made by groups such as the World Wide Web Consortium (W3C) on standards such as Cascading Style Sheets (CSS)[5], even in the case of the web where standards do exist, buggy implementations and poor uptake by web designers mean that content is still hard to extract. This restricts access to the content to its original media.

Better tools for creators of content, so they can produce documents in a medium with which they are familiar, while still taking that content and making it available for use in alternative media would be a step forward. Clearly there are some types of content that cannot be translated into other media automatically, such as photographs, so the idea of a perfect media neutral interchange medium has limitations but so far there has been little use even with textual content, despite the W3C's work in making this possible.

A trans-media scenario

Once an interchange format is established the next problem is linking this with input and output components. Input might be voice, keyboard, a pointing device of some kind, or customised switches. Output could be visual at various sizes, auditory, tactile, or some combination of all three.

To give a concrete example, imagine a scenario involving an instant messenger style application. The two participants have different limitations and therefore select different media to communicate with. One is deaf, while the other has poor vision and limited manual dexterity.

For the deaf person, their chosen input might be keyboard and mouse, since this is cost effective, and they have no difficulty using it. Their output is a standard video display. Their friend however, prefers voice input due to their limited manual dexterity, and large print video output due to their poor vision.

The deaf person's machine uses a simple input component to take commands and content via the keyboard and mouse. Their output component is also simple – it simply displays the content on screen.

Their friend has a more complex setup. A voice input component replaces the keyboard and mouse. It connects to the video output component for delivering feedback, such as prompts, which are part of the interface, as opposed to being part of the content.

With this setup the two people can communicate, both using their preferred media.

There are two major benefits to this architecture. One is that the two users are using different interfaces to what is essentially the same application. The other is that either can change their preferred medium should their needs change, and the change will be system wide.

Should the person with poor vision find that it degrades further they may find that it becomes easier for them to use spoken output instead of visual.

Different interfaces for different media

Another difficulty with systems which add to a standard windowing interface to allow improved access is that no changes are made to the design of the interface. Microsoft provides a tool called Narrator, which will speak the contents of dialog boxes. However, since the dialogs were not designed to be read they translate poorly into a vocalised interface. To be really effective the interface needs to be designed for the specific media.

The visual interface for the instant messenger outlined above might show a list of someone's contacts. Narrator would simply read that list, which could be time consuming. But for a blind user a well-designed audible interface would take an entirely different approach. It might respond to question such as "who is online?", or commands like "chat to Harry". While a video display can enumerate lists of contacts, doing so in an audible interface is tedious. Displaying a history of the conversation is also useful in the visual interface but there are serious problems in trying to translate that functionality over to a voice interface in any trivial manner. Screen readers are simply not up to the task.

This requires a stronger separation between the application, and the interface, so that the interface segment can be replaced. Applications, as traditionally constructed are tightly coupled with the interfaces they present, making it very difficult to build different interfaces without re-implementing the application code as well. While Narrator makes is possible for a blind user to use a visual interface it is a far cry from being a well designed audible interface in its own right.

The Amulet project[6] demonstrates an architecture in which the interface can be better separated from the underlying data, and where changes to the interface can be made without changing the application code. While Amulet deals only with visual interfaces a similar architecture might be applied to other interface media.

To adapt an application would therefore require making a customised interface component, which would sit between input/output components and the application itself. With interface design tools the effort required to produce this customisation could be vastly less than the effort required to produce an entire equivalent application. While the effort required might be greater than that needed for current screen reader tools the quality of the interfaces offered to end-users could be greatly enhanced.

The interface is an instance of the Bridge Pattern[7], in that it allows both the application and the input and output components to change independently of each other.

Advantages for all users

While the ability to use whichever interface a user finds most comfortable would benefit those with some impairment most, it is also very likely that able-bodied users would also gain from the ability to choose their interaction medium. Already some applications offer vocal output, for example iChat, a MacOS instant messenger application, and in some cases this can be a useful way to interact. Given a benefit to all users any feeling that those with impairments were using a 'special' system could be alleviated, since all users would use heavily customised systems which matched their own working habits.

The ultimate question is should these interfaces be considered in the category of 'special needs'? Our response is that they should not. Designing EAT systems and appropriate interfaces should not be confined or marginalised. If a system has a flaw and cannot be used for its intended

purpose, then we would suggest that the system is not dependable. If the system works in an unexpected manner or is too complex, or poorly designed they are also undependable[8].

Our current work with Methodist Care Group (part of Methodist Homes) concerns the development of a communication system that enables residents of a multi-occupancy dwelling to communicate with each other in a number of different ways. These systems are co-operatively designed with the residents themselves determining the look and functionality of the finished system. The residents have a number of different impairments and consequently the diversity of their activity patterns and ways they wish to use the tool feeds directly into the design. The residents range in age from 56 through to 96 and have differing experiences with technology. Some residents are competent to use computers and word processors, but find that many of the features of these systems are limiting due to their own impairments. For other residents, texting is a common form of communication, whilst others find using a standard landline phone difficult and limiting.

Ideally, the product of this research will be a useful tool to enable communication between residential spaces in a number of different manners. For some residents, speech is the preferred medium, where as others are familiar with a qwerty keyboard and would like to use this method of entry. Other residents would like the ability to just write freehand into the tool thus making communications more personal. Although, the residents do not, at this time want cameras attached to the system, as this might infringe on their privacy and make the communication too formal, they enjoy the idea of seeing the other person with whom they communicate. All these forms of communication are possible today, but there is no standard package that you can buy that enables all of these features. Moreover, the system is required to provide a usable interface to meet the needs of people with visual impairments as well as the inability to use interfaces with certain colours.

Conclusion

How can we make a system that is easy for everyone to use? The simple answer is to find out how people will use the system and design it with them. This paper has attempted to show that designing EAT requires a reappraisal of dependability and current UD design philosophies. We are designing for the individual, but through our designs we hope to meet the needs of a far wider audience than those we are working with. By designing interfaces that take advantage of specific input and output media, instead of retro-fitting adaptations we believe it is possible to get a superior user experience for all, and by using component technologies to do this we believe we can keep the development costs manageable.

This method of design is not new, or particularly inventive, but what is beginning to become apparent, is that if we are able to do this why are no other large companies doing this. If the need exists then let's meet it. Designing dependable 'special' systems should not be required if EAT were designed appropriately in the first place.

References

[1] DIRC http://www.dirc.org.uk/

[2] MacDonald A, “Humanising technology” in Clarkson, Coleman, Keates, Lebbon, Inclusive Design: Design for the whole population, Springer, London, 2003, pp182-203

[3] Microsoft PowerToys http://www.microsoft.com/windowsxp/pro/downloads/powertoys.asp

[4] LiteStep windows shell replacement http://www.litestep.com/

[5] Cascading Style Sheets http://www.w3.org/TR/REC-CSS2/

[6] Brad A. Myers, Richard G. McDaniel, Robert C. Miller, Alan S. Ferrency, Andrew Faulring, Bruce D. Kyle, Andrew Mickish, Alex Klimovitski and Patrick Doane. "The Amulet Environment: New Models for Effective User Interface Software Development," *IEEE Transactions on Software Engineering*, Vol. 23, no. 6. June, 1997. pp. 347-365.

[7] Gamma E, Helm R, Johnson R, Vlissides J, Design Patterns: Elements of Reuseable Object Oriented Software, Addison-Wesley, pp151

[8] Dewsbury G, Sommerville I, Clarke K, Rouncefield M (2003) ‘A Dependability Model of Domestic Systems’, in Anderson, Felici & Littlewood (Eds), Computer Safety, Reliability And Security: 22nd International Conference, Safecomp 2003, Proceedings, Lecture Notes In Computer Science, 2788, Springer Verlag, pp103-115