Supporting Context Driven Change in a User Interface

Crease, Murray

Publisher's version / Version de l'éditeur:

Proceedings of the 16th International Information Resources Management Association Conference (IRMA 2005), 2005

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
https://nrc-publications.canada.ca/eng/view/object/?id=d8519dbe-e516-4d55-ba1b-94028ebfa67
https://publications-cnrc.canada.ca/fra/voir/objet/?id=d8519dbe-e516-4d55-ba1b-94028ebfa673

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

Supporting Context Driven Change in a User Interface *

Crease, M.
May 2005

* published at the 16th International Information Resources Management Association Conference (IRMA). San Diego, California, USA. May 15, 2005. NRC 47452.

Canada

Supporting Context Driven Change in a User Interface

Murray Crease
NRC-IIT e-Business,
46 Dineen Drive,
Fredericton,
NB, Canada, E3B 9W4
Email: murray.crease@nrc-cnrc.gc.ca
Tel: +1 (506) 444-0496
Fax : +1 (506) 444-6114

ABSTRACT

This paper discusses the design of a toolkit of user interface components (widgets) that change their interaction method based on the context. The types of change supported range from relatively simple "look & feel" transformations through radical alteration of interaction modality. These changes are made dynamically and are based on resource availability (e.g., display area), global demands on the resource (e.g., the set of all current feedback requests) and context of use (e.g., ambient noise). The importance of making these changes dynamically is highlighted when considering mobile application users whose context will continually change.

The toolkit architecture is presented, focussing on the specification of widget behaviour and the mechanism for capturing and effecting context-sensitive changes to the interaction. An initial implementation of the system which offers multimodal, resource-sensitive output is described and the issues involved in extending the implementation are discussed.

INTRODUCTION

The environments in which applications are being run are often mobile and capable of being monitored via a variety of sensors. Systems that take advantage of this dynamic environment are difficult to design and build [3]. The challenge for user interface software developers is to be able to provide functionality cost-effectively in a way that best exploits the available resources. This paper presents a partial response to this challenge: a toolkit of resource-sensitive, multimodal widgets. The toolkit's widgets can use multiple forms of feedback, using the most appropriate form(s) according to resource availability and suitability. If the user is using a hand-held device then the display of large amounts of visual feedback is not feasible. Similarly, if the user is in a loud environment then perhaps the use of audio feedback is not appropriate.

Of course, the presentation of the widgets is only half of the user-interface story. Just as the resources available for presentation may vary in availability and suitability as the context of the user changes so may the suitability and availability of input mechanisms vary. This paper also considers the mechanisms required to ensure consistency between widget input and output. If, for example, the presentation area of a widget changes due to a change in screen size this has a direct impact on the input area to the widget. The proposed solution to this problem is the use of interaction spaces which define the space in which an input mechanism or output resource exist, enabling communication to take place between the two.

MOTIVATION

The toolkit of widgets described in this paper can be considered to fulfil four main objectives.

Dynamic Configuration

The widgets described in this paper are capable of modifying their input behaviour and output presentation dynamically. This contrasts with the approaches being taken by, for example, UIML [2] or Plasticity [3]. In these cases the user interface and, importantly, the platform are modelled prior to the interface being created. Once the interface has been created there is no scope for changing it according to the context. Multiple interfaces can be built using the same application model for multiple

platforms but there is no support for significant run-time change. This is significant as the environment in which the application is running can no longer be considered static when considering mobile devices.

Decoupled Decision Making

The widgets in the toolkit have no part in the process of determining how they can be interacted with. Whilst the widgets may have some pertinent information, given to them by external sources, they have no understanding of what this information represents and how it should be used. This is appropriate because it is not reasonable to expect a widget to understand its wider context - e.g. the ambient volume of the room - or whether its request for feedback is appropriate or not - e.g. the number of sounds already being played.

Exploitation of Contextual Information

The toolkit is able to exploit different strategies in using the contextual information to modify interaction with the interface. This means that simple strategies can quickly be implemented but there is no limit on how these strategies can evolve or be replaced. A sensor which detected the ambient volume could be used to adjust the volume of any audio feedback being generated. This strategy would be effective until the ambient volume increased to a level where it is no longer feasible to increase the volume and an alternative strategy would need to be employed. This strategy would not necessarily replace the simpler rule but rather a third rule would be used to determine which strategy should be applied in a particular situation.

Multiple Modalities

The toolkit described in this paper does not assume that the feedback will be in the form of graphical output. Indeed, no assumptions are made about the form of the feedback at all as the most appropriate form of presentation will depend upon the context. The context will define what presentation resources are *available* (and in what quantities) and what would be most *suitable*. A handheld device with a small screen will have less visual presentation resource available than a desktop machine with a large monitor for example. Equally, as described in the previous section, audio feedback is more appropriate in some contexts than in others.

TOOLKIT ARCHITECTURE

In this section the architecture of the toolkit is discussed, focussing on the key design-oriented features described in the previous section. An overview of the toolkit architecture is given in Figure 1.

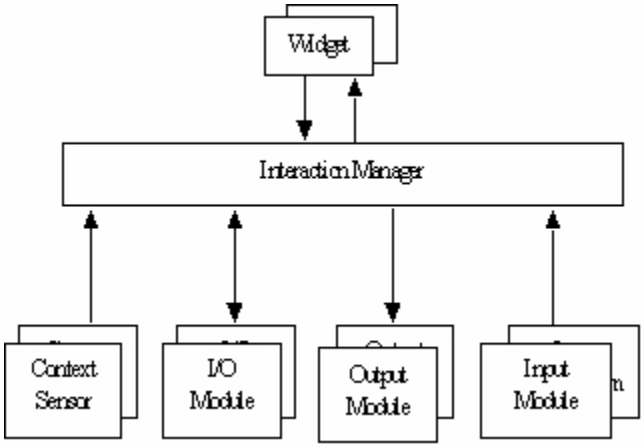

Figure 1: The high level architecture of the toolkit showing how the major software components of the system are linked.

There are three main components in the architecture: the widgets which encapsulate the abstract i/o abilities of the interface components; the interaction manager which coordinates i/o devices and widget requirements; and the external modules which provide the concrete interaction capabilities. These components are discussed in more detail in the following sections.

Widgets

The widget consists of an abstract representation of the widget's behaviour allied with some concrete knowledge of different input mechanisms and presentation forms. The widget would, for example, know that it is using a particular output module to provide feedback and that a particular set of options could be used to tailor that feedback. The widget is only the repository for that information. It does not play a role in deciding what this information should be. Each widget consists of several sub-components as shown in Figure 2.

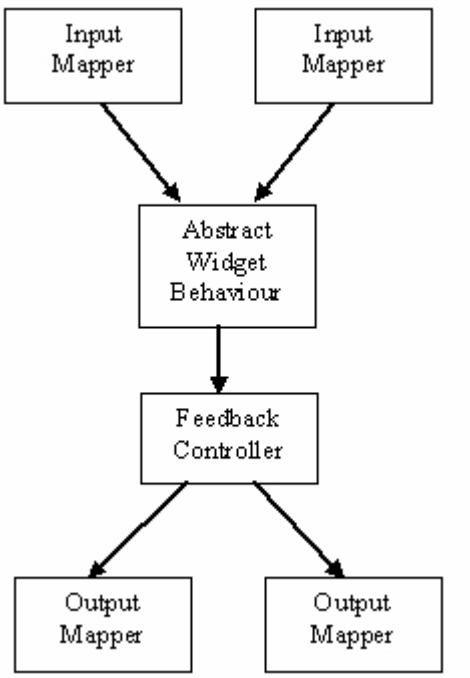

Figure 2: The components that comprise an individual widget.

The feedback controller stores the knowledge of what output modules the widget is currently using. For each output module used it creates an Output Mapper (OM) that stores information that the output module uses to tailor the widget's feedback. This information could be simple parameters or could be more explicit feedback instructions. For example, the OM could produce VRML which a VRML output module would render appropriately. In both cases, the widget is simply requesting feedback with no concept of its appropriateness or suitability, as discussed in Section 2.2

The behaviour of the widget is defined in both the Abstract Widget Behaviour (AWB) and the Input Mappers (IM). The AWB defines the abstract behaviour of the widget whilst each IM defines the concrete behaviour of the widget for a particular input mechanism.

Interaction Manager

This component is the driving force behind the toolkit's ability to manage the presentation of the widgets. It interfaces with the context (in the form of sensors and the output modules), applications (through an API) and users (via a user interface). It also makes the decisions regarding the presentation of the interface using rules which are driven by information provided by the context. The interaction manager uses two concepts which allow it to manage the presentation of the widgets: interaction spaces and rules. The interaction spaces define the space in which the widgets live and the rules determine the changes that should be made to the interface according to the capabilities and state of the interaction spaces.

An interaction space can be considered as defining the state and capabilities which a widget can use in a particular context. A windowing interaction space, for example, would define the input mechanisms (such as mouse and keyboard) and output resources (such as a visual display) available as well as an indication of the amount of resource available and its suitability. The screen may be described, for example, in terms of width and height in pixels and its suitability may depend upon the ambient brightness of the environment. An interaction space would also define the parameters within which the presentation may be modified. In the case of a windowing interaction space these parameters may include presentation parameters such as size and contrast and input parameters such as click or move over to select.

I/O Modules

The input and output modules are software components, which act as the toolkit's interface to the outside world. There are three basic types of module, although a individual module may consist of more than one type. Input modules encapsulate input mechanism(s), output modules encapsulate output modality(ies) and sensors encapsulate a part of the changing context in which the toolkit resides. Output modules map widgets' requests for feedback into concrete presentation. They can be considered as simple renderers. Input modules map user events into events that are understood by the toolkit. Sensors map changes in the context to events that can be understood by the toolkit. The events generated by sensors will often drive changes in the interaction manager's rules but may also act as an input to the toolkit's widgets.

IMPLEMENTATION

An initial prototype has been implemented in Java, based on the Java Swing toolkit (described more fully in [1]). The prototype is focussed on the modification of the presentation of the widgets according to their context. Consequently, the input side of the architecture has not been implemented and the abstract behaviour of the widget has been defined in terms of the concrete behaviour of the widget with respect to mouse input. The advantage of this approach is that it enabled a speedy implementation which proved the concept of the architecture was fundamentally sound. A disadvantage was the limited scope in changing the graphical presentation of the widgets due to the tight coupling between the graphical presentation of the widgets and Java's standard AWT event mechanism.

CONCLUSIONS

This paper describes a user interface toolkit which allows the interaction with a user interface to be dynamically changed to suit the current context. These changes are managed in terms of interaction spaces which define the techniques available for input and output; the resources available to support these techniques and suitability of these techniques in different contexts.

The current implementation of the toolkit allows for the alteration of the widget's presentation according to the context. Using the toolkit, the size and volume for the presentation of the widgets can be alerted according to the context. The future of the toolkit lies in the continued implementation so the input to the widgets can be handled as flexibly as the output is currently. This requires the concept of interaction spaces which encapsulate the resources that can be used for interaction. Doing this would provide a toolkit for quickly developing interfaces that are effective regardless of the context.

ACKNOWLEDGMENTS

Some of the work described in this paper was funded by EPSRC grant GR/L79212.

REFERENCES

[1] M. Crease, P. Gray, and S. Brewster, "A Toolkit of Mechanism and Context Independent Widgets," in *7th International Workshop , Design, Specification and Verification of Interactive Systems*, P. Palanque and F. Paterno, Eds. Limerick Ireland: Springer-Verlag, 2000, pp. 121-143.

[2] C. Phanouriou, "UIML: A Device-Independent User Interface Markup Language," in *Computer Science*. Blacksburg: Virginia Polytechnic Institute and State University, 2000, pp. 161.

[3] D. Thevenin and J. Coutaz, "Plasticity of User Interfaces: Framework and Research Agenda," in *Proceedings of Interact'99*, vol. 1, A. Sasse and C. Johnson, Eds. Edinburgh: IFIP, IOS Press, 1999, pp. 110-117.