CyberDesk: Automated Integration of Desktop and Network Services

AndrewWood

School of Computer Science
The University of Birmingham
Edgbaston, Birmingham, B15 2TT UK
amw@cs.bham.ac.uk

Anind Dey, Gregory D. Abowd

Graphics, Visualization & Usability Center
Georgia Institute of Technology
Atlanta, GA 30332-0280 USA
+1-404-894-7512
{anind,abowd}@cc.gatech.edu

ABSTRACT

The CyberDesk project suggests a way to break the prevailing assumption in personal computing that the user must search out ways to integrate behavior between separate services. We present a technique and prototype system for automatic integration of desktop applications and network services that requires no effortby either the designer or the end-user.

Keywords

Adaptive interfaces, automated integration, future computing environments, ubiquitous services

INTRODUCTION

The expectation in personal computing is that the user must search out and find the computer's interfaceto access a computational service, such as a calendar manager or an e-mail browser. Future computing environments should provide ubiquitous services that find the user by being available on any device and that are automatically integrated with a changing set of surrounding services. In this note, we describe the CyberDesk project that addresses automatic service integration.

One approach to integration is a tightly-integrated suite of tools that take advantage of known services. This approach, available in many commercial personal productivity products, is unsatisfactory for two reasons. First, it requires the designer to predict how the user will want to integrate a set number of services. Second, it forces the user either to be satisfied with what the designer has provided or to program additional and sometimes complex relationships between existing services. The CyberDesk prototype shows that it is possible, however, to provide a service integration framework that removes most of the programming burden (of designer and end-user), provides greater flexibility to the user *and* automatically suggests how two services can be integrated based on natural user input. For example, while the user is reading some mail, simply highlighting someone's name in the message can trigger the system to inform the user of all of the services available that are relevant to that name. The e-mail tool in Netscape offers similar but limited functionality in automatically recognizing URLs and e-mail addresses.

THE CYBERDESK PROTOTYPE

The current CyberDesk prototype consists of a set of personal productivity services, Java applets written by other students at Georgia Tech, and several network services, commonly used by Web surfers. The integration of these services occurs automatically based on user interaction with one of them. The user highlights some text in the window of one service, and CyberDesk determines the type of the text to suggest how the user can invoke behavior in the other services using that text. The suggestions made by CyberDesk appear as a dynamic button bar in a separate "ActOn" window.

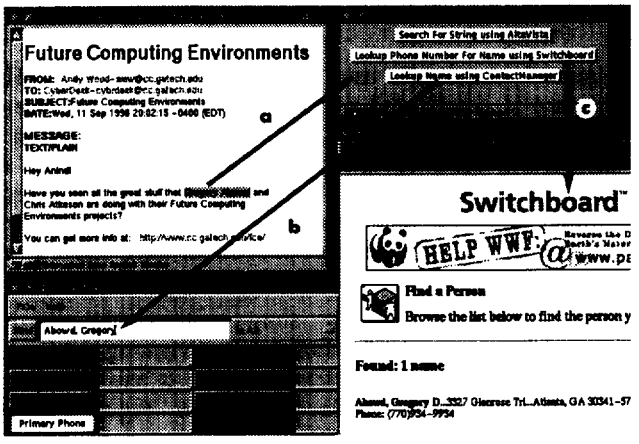

Figure 1 A sample interaction with the CyberDesk. Highlighting a name in the e-mail browser suggests several other actions that can be performed using that name.

For example, at the top left in Figure 1, is an e-mail message informing Anind about the great work going on in the Future Computing Environments group at Georgia Tech. Anind is intrigued and decides to investigate further. Highlighting "Gregory Abowd" causes the ActOn button bar to suggest some actions (a). One suggestion is to look up the name in an available contact manager (b). Anind discovers that he doesn't have Gregory's phone number, so he decides to follow another suggestion and initiates a search using the Switchboard Web service (c).

Figure 2 continues the scenario. After speaking with Gregory, Anind wants to visit Georgia Tech, but first he will do some research. He selects the first part of the URL given in the message, and the ActOn buttons change (d). Anind decides to view the URL (e) and use AltaVista to retrieve a list of Web pages that referencethe URL (f). He is pleas-

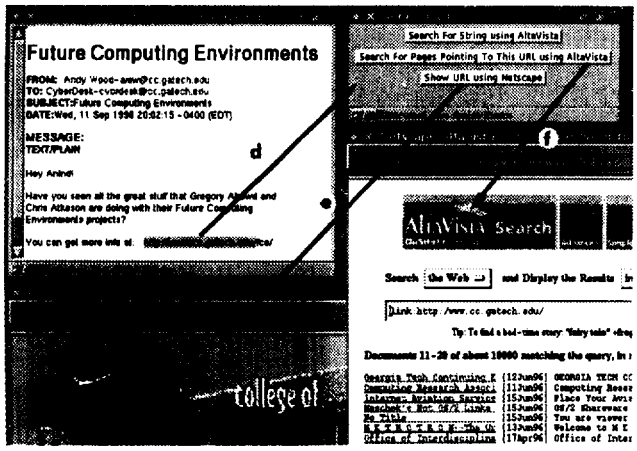

Figure 2 Continuing the scenario with CyberDesk. Selecting only part of a URL in the e-mail message suggests further integrating behavior.

antly surprised by this last option, as he was not aware that such a service even existed.

INTEGRATION ARCHITECTURE

CyberDesk services are Java applets collected on a single Web page. The applets are either local services, such as the e-mail browser and contact manager shown in the scenario, or simple wrappers around network services, such as Switchboard or AltaVista. A service can generate (display) and/or consume (accept) data of different types, as shown in Figure 3. Also included on this page are a set of type conversion components that specialize in translating generated data from one type to another. A final applet on the page provides CyberDesk's integrating behaviour: the dynamic ActOn button bar.

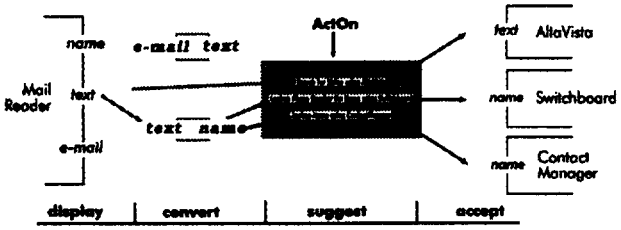

Figure 3 The run-time architecture of CyberDesk.

When the user selects information displayed by one service, say some text from the e-mail message, the type converters try recursively to see if the data can be converted to other types used in the system (e.g. a name in Figure 3). In the case of plain text, this could be done by comparing the string to common formats for representing the various types; for names you might use *title firstname lastname*, and similar patterns can be used for dates, URLs, e-mail and mailing addresses. The type converters do not have to be overly clever, as the user provides a very focused subset of the data to look at by explicitly selecting it.

Finally, the user's selection, plus any extra type information generated by the conversion process, is observed by the ActOn integrating applet and a set of potential actions for that data is suggested. For example, a name is accepted as input by the Switchboard service, and so ActOn creates a button that suggests that integrating behaviour. Clicking on the button invokes the Switchboard service, completing the integration without requiring any change to the functionality of either service and without any programming effort from the user.

ISSUES

Though we have demonstrated a novel integration mechanism for personal and network services, some system and user issues still remain. The integration scheme requires no programming by the end-user or the original designer of the service, but at this point some programming effort is required to complete the integration of a service into CyberDesk. Currently, this is simple wrapper code that informs the type converter and ActOn applets of the types the service displays and accepts. Ultimately, this service information will be automatically detectable at runtime by adherence to component software initiatives, such as Java Beans [1].

From the user's perspective, CyberDesk offers tight integration between different services, but with the freedom to introduce new services, and upgrade old services without a loss of integrating power. Integrating behavior is actively suggested by the system, removing the need for the user to remember how services work together. It is fairly clear from our use of CyberDesk that it suffers from the potential problem of having too many ActOn buttons generated; the user could be swamped by too many choices in an ever-expanding button bar. We can certainly investigate different interface representations of the button bar to help manage this. It will be more interesting to apply some intelligence to the ActOn applet to use contextual information and user history in determining the relevance of potential future actions and reduce the number of suggestions.

Another potential user problem is the reaction to a constantly changing interface. There is a clear link between our work and the adaptive user interface community. However, most of that literature concentrates on adapting a computer interface to the changing capabilities of the user, not changing capabilities of the software environment [2].

CONCLUSIONS

The CyberDesk project is a shift away from the traditional view of the desktop as a static collection of applications that the user switches between to complete a task, transferring data between them as required; in effect, chasing the required functionality through a user interface maze. Instead, our approach is to present the user with an environment in which the required functionality comes to find the user. This environment requires no extra programming burden for the designer or user of a service. It is a more flexible and useful paradigm for interaction in future computing environments.

REFERENCES

1. JavaSoft. Java Beans Homepage. Available at http://splash.javasoft.com/beans/.
2. Schneider-Hufschmidt, M., Kuhme, T., Malinowski, U. (eds.) Adaptive User Interfaces: Principles and Practice. North-Holland Elsevier Science, 1993.