Chapter XXII
Multimodal Software Engineering

Andreas Hartl
Technische Universität Darmstadt, Germany

ABSTRACT

Ubiquitous computing with its multitude of devices certainly makes it necessary to supplant the desktop metaphor of graphical user interfaces by other kinds of user interfaces. Applications must adapt themselves to many modalities: they must support a wide variety of devices and interaction languages. Software engineering methods and tools also need to embrace this change so that developers can build usable adaptive applications more easily. This chapter will present three different software engineering approaches that address this challenge: extensions to Web-based approaches, abstract user interface definitions that add a level of abstraction to the user interface definition, and model-based approaches that extend model-based application development to integrate user interface issues as well.

INTRODUCTION

Ubiquitous computing (UC) makes computing power available in a lot more devices than just PCs. Consequently, the currently prevailing methods of human-computer interaction will come under heavy pressure from other ones. The desktop metaphor introduced with the Xerox Alto and popularized by the Apple Macintosh is already being challenged by Web applications that use other ways to organize their content—usually such applications are centered around the idea of a page instead the desktop.

The Idea of Multimodality

Most prevalent forms of human-computer interaction focus on graphics and written text for interaction between users and computers. For everyday users, such user interfaces are much easier to use than their command line-based predecessors. Yet they are not necessarily the best way for human-computer interaction: the evolution of user interfaces continues and multimodal user interfaces are a promising concept for the future.

The term *multimodal* was first used by Bolt (1980) in his paper "Put-that-there." With multimodal user interfaces, users interact with the system using several independent means of interaction. "Put-that-there" used graphical output and speech and gesture input and eye tracking. Colloquially speaking, one could define multimodality as "multimedia the computer understands." Nigay and Coutaz (1995) provide a more formal definition; to them, a modality is a *coupling of an interaction language L with a physical device d: <d, L>*. The formal definition of a physical device is that of *an artifact of the system that acquires or delivers information. An interaction language is a language used by the user or the system to exchange information. A language defines the set of all possible well-formed expressions [...].*

We will use this definition of a modality throughout this chapter.

Windows, Icons, Menus, and Pointing Devices and Beyond

The first devices that challenged the personal computer's dominance as the most used tool for accessing and processing information were cellular phones and personal digital assistants like the Palm or the Blackberry. As means of interaction with their users, these small devices continue to use the established graphical interaction techniques based on windows, icons, menus, and pointing devices, albeit slightly modified. Application developers know how to build user interfaces with such techniques, which makes it easy to adapt similar techniques also for mobile devices. Yet the widening range of possible target devices will probably change the user interface from slightly modified desktop interfaces to something new for several reasons:

- Graphical user interfaces implicitly define some constraints on the devices that are using them: they require a decently sized display, and a way to move the pointer, two things that may not be available on a very small device.
- By their very definition, graphical user interfaces focus on graphical output and input, often neglecting other possible means of interaction. Hence, users can only interact with such an interface if they are not too distracted and if their hands and eyes are free to use it.
- Interaction metaphors that work well for desktop-based graphical user interfaces may not be adequate for UC devices. Other forms of interaction may be better suited for ubiquitous computing devices. Satyanarayanan (2005) gives Apple's iPod as an example, whose "success in displacing older portable music devices shows the power of combining the right device functionality with a good form factor and user interface."

Most applications apply a closed-world approach where every property of the system is known beforehand. For example, applications know the minimum screen size they can expect, and input and output features available to them. This makes it possible to tailor the user interface well to specific devices, and ideally makes an application easier to use. The drawback of this approach is that the application cannot adapt its user interface beyond the closed world it lives in, even if that would make sense; for example, it cannot use a newly found large screen for displaying complex data but stays restricted to the small screen of the portable device it was developed for.

As the number of devices that want to communicate with their users increases, the closed-world approach becomes less and less feasible for each of them. The most pressing issue is of