Chapter XIX

Adaptive Interfaces in Mobile Environments: An Approach Based on Mobile Agents

Nikola Mitrovic
University of Zaragoza, Spain

Eduardo Mena
University of Zaragoza, Spain

Jose Alberto Royo
University of Zaragoza, Spain

ABSTRACT

Mobility for graphical user interfaces (GUIs) is a challenging problem, as different GUIs need to be constructed for different device capabilities and changing context, preferences and users’ locations. GUI developers frequently create multiple user interface versions for different devices. The solution lies in using a single, abstract, user interface description that is used later to automatically generate user interfaces for different devices. Various techniques are proposed to adapt GUIs from an abstract specification to a concrete interface. Design-time techniques have the possibility of creating better performing GUIs but, in contrast to run-time techniques, lack flexibility and mobility. Run-time techniques’ mobility and autonomy can be significantly improved by using mobile agent technology and an indirect GUI generation paradigm. Using indirect generation enables analysis of computer-human interaction and application of artificial intelligence techniques to be made at run-time, increasing GUIs’ performance and usability.
INTRODUCTION

Mobile computing is an increasingly important topic in today’s computational environment, because the demand for ubiquitous access to information is constantly increasing. Furthermore, users want to increase their efficiency and process information when using mobile equipment. To support this demand, software applications face a number of challenges. One of the important challenges in mobile computation is user interaction.

The importance of user interfaces (UI) comes from the fact that UIs represent the first line of interaction between a user and a computer. A user’s ability to execute a required task and his efficiency are directly impacted by the user interface.

In the past, user interfaces have been developed mostly for a specific device, for example a specific PDA variant or a work station. Such an interface was usually designed for a single platform. This was done in conjunction with specialized user interface libraries that were defined for a specific platform or programming language. For example, if we assume an application that is developed for Windows and UNIX platforms, the user interface for the windows platform would be designed and developed separately from the UNIX user interface. The cost and effort required for such a development are obviously high; such an approach frequently leads to other problems, for example GUI implementations on one (or more) platform(s) being at different levels of development due to the lack of resources required to maintain the same GUI version on multiple platforms.

In the mobile environment GUIs face additional challenges: a user could be using an application on a mobile phone and could require the same application on his PDA or WebTV. In addition, the user could be moving and requiring an application to move with him. For example: while in the car, a user could read his e-mail using car’s on-board computer; when he steps out of the car he could prefer to continue working on his PDA until he gets to the office, where a desktop PC could be his preferred equipment to continue working.

Mobile devices have different capabilities and requirements: different processing power, screen size, supported colors, sound functionalities, keyboard, and so forth. In addition, mobile devices use an ever-increasing number of different hardware and OS solutions, and frequently rely on batteries for operation. Mobile applications use wireless networks; wireless networks are not stable, have limited capacity and performance, and are expensive (e.g., 3G networks).

In addition to this, application interface and functionality may change depending on a user’s context. For example, a music player application should mute if the user is indoors and should turn on when outdoors. Furthermore, a user could prefer the speaker to be on a louder setting when in the car. These requirements could be either a user’s preferences or rules associated with a particular location where the user is.

To meet such challenges, researchers in the user interface area have adopted a common approach—user interface abstraction. To be presented on a concrete platform, abstracted user interfaces are transformed and rendered to meet a concrete platform’s requirements. This approach provides a single user interface definition that is later transformed to the target device’s user interface. The abstraction level in such an abstract user interface definition varies. Some abstract user interface notations offer very abstracted descriptions of user interfaces, while others are more linked to specific user interface concepts, for example window-based user interfaces. An abstract user interface definition is usually delivered in XML (W3C, 2000) notation, which enables efficient processing and data exchange between multiple platforms. Some notations describe the user interface at a high level, for example, a button is required; others allow sophisticated definitions of constraints and additional parameters, such as requiring the button for some (specific) device(s) only.

This chapter presents different approaches to adapting user interfaces to devices, with specific interest focused on enabling architectures that adapt to users’ preferences and contexts. We discuss difficulties with mobile user interface generation for wireless devices. Finally, we present an approach for user interface adaptation based...