REQUIREMENTS FOR DISTRIBUTED USER-INTERFACES IN UBIQUITOUS COMPUTING NETWORKS

Erik Berglund¹ and Magnus Bång²

¹Department of Computer and Information Science, Linköping University, Sweden
eribe@ida.liu.se
http://www.ida.liu.se/~eribe

²Proxigo AB, Linköping,
magnus.bang@proxigo.com
http://www.proxigo.com

Abstract. Ubiquitous computing requires new software paradigms that allow the distribution of the user-interface among several interaction devices in the physical environment. However, current user-interface approaches such as Java’s Abstract Window Toolkit are designed from a single workstation perspective. Distributed user-interfaces allow applications to distribute, negotiate, and allocate its user-interface among variable sets of interaction devices, services, and users. In this paper, we discuss the basic requirements mobile ubiquitous computing applications place on distributed user-interfaces. We provide an overview of current trends in the area, provide user scenarios, and discuss the need for a new user interface model. In particular, the paper provides a set of requirements that underlying software architectures must meet to support developers in designing robust and efficient systems for distributed user interfaces.

Keywords
Distributed User-Interfaces, Ubiquitous Computing, Software Architectures

1 Introduction
The vision in Ubiquitous Computing is that people will interact with numerous devices that are interconnected and physically distributed in the environment [30]. This class of computer systems requires new types of user-interface approaches. Distributed user-interfaces [6] are user-interface that distributes its components among several interaction devices available in the environment. Technically, such DUI components are treated as network-entities of self-configuring peer groups that negotiate UI responsibilities.

From a workplace research viewpoint, these developments are important. For example, researchers have emphasized the importance of our physical environments and the role everyday artifacts play in supporting cognition and collaboration in the workplace [2, 3, 8, 11, 17, 18, 23]. Needed are computer-based tools and software architectures that support mobility and the physical dimension of work.

This paper discusses the requirements we can set on distributed user-interfaces from an interaction standpoint and how these requirements affect the underlying software architectures. The specific aim of the paper is to provide a set of requirements for software architectures that directly support the use and development of distributed user-interfaces.

2 Trends in user interface research
Recent developments within Computer Science set new requirements on the core architectures for interaction with computer technology. For example, Ubiquitous Computing [30, 22] refers to the state when computing power is embedded in everyday artifacts rather than in stationary computers. This trend is also observable in the development of new platforms for wireless IP [26] such as WLAN and Bluetooth that provide interconnectedness among IP-enabled artifacts and computers. Clearly, the convergence of the computer, the network, and our everyday artifacts is underway [9, 12, 29].

An example of this movement is the Anoto-system [1] where a digital pen and paper function as the interface to computers. Technologies like the Anoto-system, new sensor technologies, and embedded devices are likely to change the way we interact with computers; from the desktop computer with its QWERTY keyboard to everyday objects as interaction devices to computer resources and services in networked environments.

Workplaces of the future will be full of IP-enabled
interaction devices like displays, digital pens with their paper applications, and specialized interaction devices. Users can move seamlessly across a landscape of multiple displays and interaction devices and still access their own virtual desktops. The interaction devices allow natural and mobile work, such as paper-based work, but provide still the power of the traditional computer. In these ecosystems of interaction devices the interoperability among the devices are of importance. For example, when a user approaches a wall-mounted display with a digital pen and a paper-form, it is crucial that the combined interface (i.e., the display, the digital pen, and the paper-interface) form a functional interface unit dynamically. Thus, system needs to be aware of the interface resources in the physical environment.

The network is rapidly connecting every single computing machine and interaction device to a common network structure, enabling transparent combination of components over the network. Programming is abstracting away the physical layer and replacing it with URIs and self-configuring resources networks. The Java Jini [24] and JXTA [10] projects are examples of net-centric projects moving network programming from client-server to peer-to-peer technologies [7, 19]. Microsoft also focus on network programming in their .Net program to abstract away from the computer and towards the global construction of software applications from networks resources [15].

Before we discuss the requirements that are set on the user-interface architectures in the network-based ubiquitous computing world, let us describe two scenarios on how such an interfaces might function. Figure 1 illustrates the principle interaction components in these scenarios.

### 2.1 Working with mobile phone and fixed display

Lisa Eriksson is visiting a trade fair. She walks though the exhibition floor passing different booths. Simultaneously, she is surfing an exhibition web site with her mobile phone. As she approaches a booth in the exhibition area, her mobile phone presents the available multimedia and information services for that particular booth. The mobile phone provides a digital video clip, which she activates, and it starts playing in the mobile-phone display. However, the presence of a 50 inch screen is also detected and gets plugged into Lisas mobile web browser across the network. The available interface components compare their capabilities with regards to screen resolution and size and renegotiates responsibilities. The decision is reached to redistribute the digital video clip to the 50 inch screen while presenting a more advanced remote control GUI on the mobile phone display. As Lisa walks away, the video clip is removed from the 50 inch screen and starts playing in her mobile phone again. She stops the clip and leaves the booth.

### 2.2 Working with paper-interface and display

It’s a late Saturday evening in the local hospital. Arne, the head nurse, is working at Ward 12 and at the Emergency Room (ER). He has to walk the long corridor several times this evening down to the ER. Arne has just given a normal dose of Ketogane, a pain killer, to a patient at the ward. He must now record this fact in the medical journal of the patient. In his pocket he has a pad of frequently used digital paper-interfaces that address different tasks in the paper-based health-care system. He takes a paper interface named “Record Drug” and fills in the name of the drug and dose on the form as he walks to the emergency room. In the corridor, he approaches a wall-mounted display. The display identifies Arne and shows the ward’s virtual desktop with virtual patient folders. Arne identifies the patient’s virtual patient folder and activates it by pointing with the finger on the display. Subsequently, he takes the paper-application and starts it by checking the “start application box” with the digital pen. Over the network, available interaction components are grouped and start the negotiation for user-interface responsibilities. Available are the digital pen, the paper application, and a touch-screen display. The paper form becomes the component for data input and the display activates a feedback form and a context-dependent menu list associated with the particular form. The drug name and dose is already present in the fields (showed by the display feedback component). However, Arne notices that the dose is incorrect (perhaps a mistake on the part of the handwriting recognition software). To correct this error, Arne clicks on the display (to activate the field) and rewrites the dose on the paper-application with the digital pen. The system now identifies the correct dose and shows it as plain text in the dose-field.
3 Needed: a new user interface model

Mobile devices currently provide limited means for interaction. For example, mobile phones have, and probably will continue to have, small screens and limited input and output capacity due to the preferred form factor. Devices that can provide rich interaction modalities such as large displays are however seldom mobile. Furthermore, other interaction devices like digital pens and paper-interfaces also meet similar problems in user interface. These systems have limited output capabilities for feedback and it is therefore desirable to redirect output to screens.

Small devices such as mobile phones require the distribution of the user interface across interaction devices in the surrounding environment to provide rich interaction. Furthermore, many work situations require collaborating users to easily share their computing tasks and thus the distributed user-interface is in these circumstances a natural model of interaction. Current graphical user interface (GUI) models are designed for the desktop perspective in which users interact with software applications from one computer and one predefined set of interaction devices. GUI models such as Java Swing [25, 28], Microsoft Foundation Classes [14, 20], and Motif [16] are all designed for single workstation.

New models for the distribution of the interface across different interaction devices must be considered as services on a network that appear and disappear during a computing session. These services must dynamically form peer-groups to handle the interaction needs of an application as a functional whole. In short, mobile ubiquitous applications function in networks of users that use networks of interaction devices to complete networks of computing tasks. As application developers start taking advantage of the new abilities of ubiquitous peer-to-peer networks these new possibilities will start to work to their advantage. Though, at first hand this network complexity may seem to confusing, we believe that in many application domains where the desktop computer is a difficult tool may benefit dramatically from a distributed user interface model. From a software application perspective large changes happen as development of user interface focus on creating functional wholes of distributed systems rather than to connect separate interfaces over the network.

4 Requirements

Mobile ubiquitous applications face a completely different user than traditional desktop computing. Naturally, the new user interface model must be matched by the underlying software architectures to support adequately both the use and the development of these systems. Needed are software architectures, development tools and process, and software component libraries (APIs) that developers can use to develop robust systems effectively. The following requirements set the technical stage for distributed user-interfaces:

- Users work from different interaction device configurations that are spread over large physical spaces. User interfaces must therefore be dynamically constructed and distributed among network-connected interaction components.
- Users will sometimes have multiple similar interaction components with the same interaction capabilities, for instance a mobile phone and a keyboard (both with alphanumeric input capabilities). The peer-group must then negotiate among devices to appropriately distribute the interface and make best possible use of the available resources.
- User will, at other times, have less than optimal interaction components and therefore the distributed user-interfaces must compensate and provide graceful degradation of interaction.
- Users share interaction components with other users, requiring negotiation of interaction resources to be handled dynamically.
- Manual configuration is highly undesirable, such as logging on, connecting interaction devices, and configuring user interfaces. Reconfiguration and distribution of the interaction device group should be transparent for the user (unless required specifically by the application for instance for security reasons) and allow users to access the same user interface state from different locations.

To address these problems research in software engineering is needed in the following areas. First, semantic descriptions of interface components must be developed, standardized, and combined with plug and play network architectures. Plug and Play architectures have been used to identify devices on a lower level and to load drivers in computers [21]. Plug and play architectures also exist to identify devices over networks [27]. However, the plug and play approach does to provide support for identifying the interface resources available in the physical environment.

Furthermore, distributed layout algorithms and layout engines must be developed that treat user-interfaces as variable network-connected component peer-groups. Contemporary layout algorithms combine content with style specifications to determine how content should be displayed. An example of this approach is cascading stylesheets [5], which is the primary layout specification.
language for the web. However, contemporary layout methods do not provide a language for describing how different components should be combined and used jointly in a ubiquitous computing world.

Moreover, distributed user interface programming tools and models must be developed. GUI toolkits must be augmented or changed to provide maintainable programming models for multiple concurrent versions of user interface design. Contemporary GUI toolkits such as Java Swing [24, 28], Microsoft Foundation Classes [14, 20], and Motif [13, 16] does not support the distribution of GUIs across multiple devices other than through remote procedure calls to separately developed application GUIs.

5 Conclusion
This paper provided a starting point for research into the interaction models and for the underlying software component systems that aim to support the distributed user-interface approach. The goal is that these software architectures shall provide sound and efficient ways for developers to design distributed user-interfaces for mobile ubiquitous applications. It is clear from our analysis that the current model of users reflected in contemporary graphical toolkits and languages does not provide adequate support the new area of ubiquitous computing applications. This paper provided the basic direction for development that is needed to handle users that interact with applications in interaction device landscapes providing varying degrees of interaction granularity and doing so concurrently with other users.

6 References


