Towards Context-Aware Cross-Device User Interfaces in the Wild

Fabio Paternò, Giuseppe Ghiani, Marco Manca
CNR-ISTI, HIIS Laboratory
Via Moruzzi 1, 56124 Pisa, Italy
fabio.paterno@isti.cnr.it

Abstract
In recent years a number of frameworks for easing design and development of cross-device user interfaces have been put forward, mainly in research contexts. In general, they provide support for connection management, data synchronization, and user interface distribution. However, the applications that can be obtained can be accessed from a wide variety of contexts of use that vary in terms of available devices and connectivity, surrounding environment, user preferences and abilities, and social relationships. Thus, one of the main limitations in the adoption of such applications in the wild is the difficulty to customize them for different needs in such diverse contexts. This position paper indicates and discusses the issues that should be addressed for this purpose and intrdouces possible approaches to solve them.

Author Keywords
Cross-device user interfaces, Context of use, End-user development.

ACM Classification Keywords
H.5.2 [Information interfaces and presentation (e.g., HCI)]: User Interfaces - Input devices and strategies.

Introduction
The increasing availability of various types of devices in our daily life is often a missed opportunity since current applications are limited in supporting seamless task performance across them. Users often perceive device fragmentation around them rather than an ecosystem of devices that supports their activities. In order to address such issues a number of frameworks, platforms, and authoring environments have been proposed, mainly in research environment. The goal is to facilitate design and development of multi-device user interfaces. We can distinguish various types of multi-device user interfaces depending on the features that they support: *migratory user interfaces* are able to

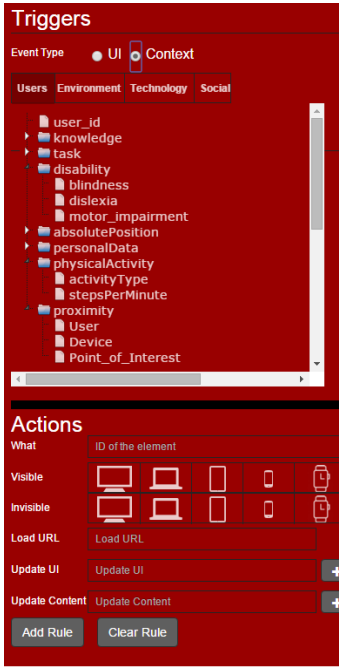

Figure 1 Tool for editing trigger action rules

dynamically migrate from one device to another in order to follow users' movements while preserving their state; *distributed user interfaces* allow users to interact with an application through multiple devices at the same time; *cross-device user interfaces* are distributed user interfaces, with the additional capability to synchronise their state, so that the interactions through some element in one device update the state of the corresponding elements (if any) in another device. Such categories are not mutually exclusive, so for example it is possible to have user interfaces that are both migratory and cross-device.

Cross-device Frameworks and Authoring Environments

In recent years some frameworks that provide useful support for developing cross-device user interfaces have been proposed. The proximity toolkit [5] simplifies the exploration of interaction techniques by supplying fine-grained proxemics information between people, portable devices, large interactive surfaces, and other non-digital objects in a room-sized environment. We have designed a framework supporting user interface distribution in multi-device and multi-user environments with dynamically migrating engines has been proposed [2]. It does not require a fixed server to manage the distribution. The elements of the UI can be distributed by specifying specific device(s), group(s) of devices, specific user(s), and groups of users according to roles. Panelrama [7] is a solution able to categorize device characteristics and dynamically change UI allocation to best-fit devices. For this purpose, this framework lets developers to specify the suitability of panels to different types of devices. The increasing use of wearables in the context of cross-device user interfaces has been addressed by Weave [1], a framework for developers to create cross-device wearable interaction by scripting. It provides a set of JavaScript- based APIs to easily distribute UI output and combine sensing events and user input across mobile and wearable devices. Other cross-device frameworks involving smartwatches have been proposed (e.g. [4]). In addition to frameworks, also some authoring environment to ease the development of cross-device user interfaces has been proposed. An example is XDStudio [6], which supports two complementary authoring modes: simulated and on-device. In the former mode, authoring is carried out on a single device in which the user interfaces distributed are simulated. In the latter mode, design and development actually takes place on the target devices themselves. However, this type of authoring environment does not provide support for specifying context-dependent behavior. This aspect has been addressed by our context-aware authoring environment [3], which supports development of user interfaces able to adapt to the various types of contextual events (that can be related to users, devices, environments, and social relationships), with the possibility of distributing the user interface elements across multiple devices. The context-dependent behavior is modelled through trigger / action rules (an example tool for editing them is in Fig.1), and can even be applied to extend the capabilities of Web applications that were not originally designed to be context-aware.

An Architecture for Context-aware Cross-device User Interfaces

In order to correctly execute the applications according to the adaptation rules specified it is necessary to have a specific architectural support at run-time. The main goals of such support are to manage and apply the user

interface adaptation or distribution rules, and detect the events that trigger their performance. Such run-time support exploits the functionalities of three components: the context manager, the adaptation engine, and the distribution manager. The context manager is composed of a context server and a set of external modules delegated to monitor relevant parameters of the context of use (e.g. environmental noise, device coordinates, user physical activity).

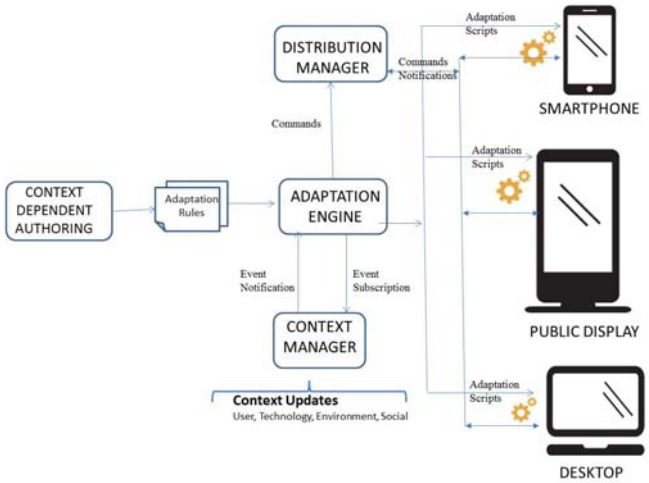

Figure 2 Architecture for Context-aware Cross-device User Interfaces.

The purpose of the context manager is to detect contextual events and inform the adaptation engine, which stores and manages the contextual rules, and requests changes in the cross-device user interface according to the triggered rules. The distribution manager handles user interfaces distributed across multiple devices in order to allow dynamic migration of components and keep their state synchronized. Figure 2 shows how such components interact with each other. The adaptation engine subscribes to the context model manager in order to be informed of the occurrence of the events relevant for the rules associated with the active applications. When one or more of such events occur, the adaptation engine sends the actions to the applications in order to perform the corresponding changes. Such update commands are interpreted by the scripts included in the application by the authoring environment. They can modify properties of user interface elements or content, activate functions or navigation, and change the distribution of some user interface parts across devices. In the latter case the adaptation engine can directly send the corresponding command to the distribution manager, which notifies the involved devices. Such distribution manager contains the current distribution profile, which indicates how the various parts of the user interface are currently distributed across the devices that have subscribed to the environment. A distribution command mainly determines whether a user interface element or the elements included in a container should be visible or not on one specific device or a group of devices that have the same role or on all devices of a given platform.

Issues for Deployment in the Wild

The approach to context-aware cross-device user interfaces is general and can be deployed for a wide variety of applications (for example smart retail, museums, smart cities, e-learning, ...). For this purpose various aspects should be considered.

Interoperability.

We need the possibility to operate on various types of devices (smartwatches, smartphones, tablets,

desktops, public displays, ..) from various vendors. Only Web applications can be accessed through almost all of them with limited effort. However, the run-time supporting the cross-device user interfaces should be able to work even when network connections to remote external Web servers is not possible (a possible solution is described in [2]). This means that the underlying architecture should be able to create peer-to-peer organization amongst the involved devices.

End-user development

In the end only the users know the best way to configure their cross-device user interfaces in their specific contexts of use, thus we need to provide them with authoring environments and customization tools that allow them to directly specify the contextual rules even if they do not know how the underlying technology works. For this purpose the use of subset of natural language to indicate the desired behaviour with familiar, domain-dependent terms can be effective.

Flexibility

The control on the cross device user interface by developers and users should be able to address various granularity levels when allocating or dynamically changing which user interface parts should be in each device. We can identify four possible granularity levels: some distribution changes can involve the entire user interface, others can only involve groups of elements, or be limited to single user interface elements (e.g. a list or a text input), or even parts of single elements (e.g. their prompt or feedback).

Modalities

Some approaches only consider graphical cross-device user interfaces but natural interaction can be achieved if the associated environments are also able to support other various modalities that users can exploit depending on the context (vocal, gestural, graphical, …) in an integrated manner.

Mixed-initiative Triggers

The changes in the configuration of the cross-device user interface can be made on explicit request through customization tools or triggered automatically by the context-dependent rules. In the latter case it is still important to make users aware when the changes occur, with also the possibility to reject them if they are not deemed useful at a given time.

References

1. Chi, P. and Li, Y. 2015. Weave: Scripting Cross-Device Wearable Interaction. CHI 2015, ACM
2. Frosini, L. and Paternò, F. 2014. User Interface Distribution in Multi-Device and Multi-User Environments with Dynamically Migrating Engines. Proceedings of EICS 2014, ACM, pp. 55-64.
3. Ghiani G. Manca M. Paternò F., Authoring Context-dependent Cross-device User Interfaces based on Trigger/Action Rules, In MUM2015, pp. 313-322.
4. Houben, S., and Marquardt, N. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. Proceedings of CHI 2015, ACM, pp. 1247-1256.
5. Marquardt, et al.. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. UIST 2011, pp. 315-326.
6. Nebeling, M., Mintsi, T., Husmann, M., Norrie, M. C. 2014. Interactive development of cross-device user interfaces. In CHI 2014, ACM, pp. 2793-2802.
7. Yang, J. and Wigdor, D. 2014. Panelrama: enabling easy specification of cross-device web applications. In Proceedings of CHI 2014, ACM, pp. 2783-2792.