A Logical Framework for Multi-Device User Interfaces

Fabio Paternò, Carmen Santoro
CNR-ISTI, HIIS Laboratory
Via G. Moruzzi 1, 56124 Pisa
Italy
{fabio.paterno, carmen.santoro}@isti.cnr.it

ABSTRACT
In this paper, we present a framework for describing various design dimensions that can help in better understanding the features provided by tools and applications for multi-device environments. We indicate the possible options for each dimension, and also discuss how various research proposals in the area are located in our framework. The final discussion also points out important areas for future research.

Keywords: Multi-device User Interfaces; Logical Framework; Distributed and Migratory User Interfaces.

ACM Classification Keywords
H.5 Information Interfaces and Presentation;
H.5.2 User Interfaces

General Terms: Design, Human Factors.

INTRODUCTION

Nowadays, it is extremely common to see users performing their tasks using various devices ranging from the traditional stationary desktop platform to mobile devices with various multimodal interaction resources. However, by now, users' expectations have not yet been adequately fulfilled. Often all this technological offering is not exploited as it could be, and when users perform cross-device service access they encounter various usability issues: poor adaptation to the context of use, lack of coordination among tasks performed through different devices, inadequate support for seamless cross-device task performance. For example, one potential source of frustration for users is the inability to continue to perform their tasks when they have to move about and change the interaction device. In such cases, users either have to manually perform some activity in the first device in order to save the up-to-date interaction state and then reconstruct it afterwards on the new device, or, in the worst case, they have to start their activities over again from scratch when moving to the second device. A study reported in [4] aimed to achieve a better understanding of why and how people use multiple devices in their everyday life. The authors found out that users already employ a variety of techniques for accessing and managing information across devices. However, there is still room for improvements, especially from the user experience viewpoint: participants in the study reported that managing information across devices is the most challenging aspect of using multiple devices. To this end, various approaches are possible. In distributed User Interfaces (UIs) we have solutions that allow users to exploit user interfaces distributed across multiple devices at a given time to access their applications. In migratory UIs users can change device and still access the application with some level of continuity, which means that at least some parts of the original user interface preserve their state after changing device. More generally, it is important to reach a better understanding of how we can design tools and applications exploiting multi-device UIs, which is the main goal of the framework that we propose.

In this paper, after discussing some relevant work in the area, we first suggest a logical framework in order to describe the range of possibilities that multi-device UIs offer, by identifying ten dimensions that have been judged relevant for such systems based on our analysis of the state of art and experience in designing multi-device environments. Then, we summarise the main points of the framework by also providing a table supporting an analysis of some proposals in this field. Finally, we conclude with some summary remarks and indications for areas that are currently underexplored and need more research work.

STATE OF THE ART

The *recombinant computing* approach [7] has been proposed and investigated to facilitate users in exploiting multiple technologies in a composite manner (rather than in isolation). On the one hand, the approach does not require the involved components to have mutual awareness, but on the other hand the components have to specify how they exchange information (thus, they need to have a recombinant implementation). A recent study [17] aimed to investigate the key elements that characterise the User eXperience (UX) when users exploit Web-based applications through different computing platforms (mainly desktop and mobile devices). This study also identified an initial framework for cross-platform service UX, in which the central elements include i)fit for cross-contextual activities (the structure of the application across different devices matches the user's activity, leading to an effective fit for tasks in different contexts), ii)flow of interactions

EICS'12, June 25–26, 2012, Copenhagen, Denmark.

and content (the transitions across the devices are experienced as fluid and connected), and iii)perceived service coherence (the application and its components are perceived as consistent and coherent, as part of the same service). In our paper we aim to provide a more structured framework, which is also able to highlight the main technical issues in multi-device User Interfaces (UIs).

Demeure et al. [5] have investigated distributed user interfaces (DUIs). The introduced reference framework defines four possible dimensions for the distribution: what is distributed, who is distributing it, when, from/to where. According to the authors, the framework does not consider the collaboration among users since this dimension is assumed to be "a natural extension" of DUI when several users are involved in the distribution. Some analogies can be found between that framework [5] and ours: the computation dimension, which introduces the notion of splittability, defines which parts of the interface can be distributed and is similar to our "granularity"; coordination is related to the "trigger activation type"; communication corresponds to "UI generation phase". However, the main difference with our proposal is that our framework addresses multi-device UIs in more general terms (including also e.g. as resulting from migration) and is not limited to DUIs.

A problem space for UI plasticity is proposed in [3]. The problem space is defined by a few dimensions and an interactive system is modelled as a graph of models that can be dynamically manipulated by, and/or encapsulated as services. Their approach to this problem is to bring together MDE (Model Driven Engineering) and SOA (Service Oriented Approach). In our case, we provide a more detailed set of dimensions in which the tool support for multi-device environments can also be analysed, and we do not limit our analysis to model-based approaches.

Myngle [15] is a support that facilitates device change (e.g. desktop to mobile) in Web navigation. It provides an easy way to revisit content previously accessed, by providing a unified web history from multiple personal devices, and allowing users to filter their history based on high-level categories. Since it is proposed as a browser extension (Firefox, Chrome) and as a native application for a few mobile platforms, thus it lacks portability. Also, it cannot support access to resources (e.g. session cookies, JavaScript variables, …) that are not usually mapped in the URL.

Multi-device applications are important in many domains. In [9] the authors investigated how to improve learnability in ubiquitous systems (e.g. multi-device museum guides). Authors claim that one design principle to promote the learnability of such systems is to improve their UI consistency. They identified three types of consistency, in order to better address the issues of multi-device, multi-user contexts: i)*within-device consistency*, which occurs when the UI design is consistent with the design of previous applications developed for a specific mobile device; ii)*across-device consistency*: it occurs when the UI design is consistent with the design developed on other mobile devices; iii)*within-context consistency*: when consistency is applied to the context, including aspects that are not strictly connected with the devices. For instance, a museum is a social, informal context in which interactive learning is supported: therefore, the user interface should reflect these characteristics to some extent. To this goal, the authors have compared three different interaction styles (gamepad controller emulation, mobile multi-touch, and Wii-based emulation) and they concluded that the within-device consistency generally gives better results for museum applications (these results can be also generalised to other similar ubiquitous contexts).

Previous work by the research community has coped with multi-device access to applications. Olsen et al. [13] studied techniques to combine multiple clients working on the same task, and have introduced the concepts of "Join" and "Capture". *Join* refers to collaborations with other users, whose clients can subscribe to data associated to a particular task; clients are promptly notified whenever data change. *Capture* consists of assembly of interactive resources to address specific problems, for instance by exploiting different modalities, with the aim to improve interaction. The authors claim that their integrated approach of multi-client interaction and multi-user collaboration is able to provide synchronization among tasks. This aspect is related in some way to the migration dimension of our framework. However, the paper does not consider a scenario where multiple users, independently from their physical position, share application interfaces. The authors also state that, although the multiple devices involved share the same network and are integrated in the same task, they are not aware of each other.

In general, we found a variety of research contributions in the area of multi-device user interfaces, and we cannot mention all of them. However, despite the increasing interest on this topic, when designers and developers want to support users in accessing applications (or parts of them) through various devices, they have often difficulties in identifying the possibilities and aspects that should be considered. The main goal of our framework is to provide a set of dimensions that allow users, designers, and developers to analyse the possibilities of tools and applications accessed in multi-device environments. This is useful both in the design of new multi-device environments and in the evaluation of existing ones.

A LOGICAL FRAMEWORK FOR MULTI-DEVICE UIs

A number of factors enable the analysis, design and comparison of multi-device user interfaces. Such factors have been identified by analysing related work in the area, emerging applications, and carrying out research in tools for supporting multi-device environments.

There are many ways to support multi-device applications. They range from accessing them through different devices at different times (one device at each time), to situations in

which users access multiple devices at the same time through a UI that dynamically changes its distribution across them. Other cases could occur when the user changes the interaction device and the user interface supports (or not) the possibility to preserve the interaction state. Thus, with this framework we want to systematically analyse the various possible situations in which cross-platform access can be performed. These dimensions are described in the following subsections in which we consider various types of multi-device UIs. A solution can assume just one value in each dimension (when applicable). We have not identified particular dependencies between dimensions and the range of the possible values is specified for each framework dimension.

UI Distribution

This aspect analyses whether the solution considered is able to support the distribution of the user interface elements across various devices at a given time. We have distributed UIs when the UI elements of a given application are distributed across more than one device. In such distribution some elements can be even duplicated. Therefore, since at each time there are (at least) two devices involved in the rendering of the UI, UI distribution implies the existence of some coordination across the involved interactive devices supporting the access to the application logic by exploiting input/output from/to the various devices involved in the distribution. An example is when people access large screens to see large amount of information and use a mobile device to enter some queries. As for the range of values, the distribution can be *dynamic* (when the user interface elements can vary their allocation to the devices during a user session) or *static* (when the distribution configuration cannot change during a session).

An approach for *dynamic* distribution of UIs at run-time is discussed in [1], where the authors highlight the availability of diverse devices that characterise smart networked environments. Among the potentialities of distributed user interfaces in such contexts, there is the possibility for the user to enhance the interaction, for instance by increasing the communication bandwidth.

Although not mentioned explicitly as distribution, an early example of interface splitting among several devices was also tackled in [12]: Pebbles SlideShow Commander allowed users to control a PowerPoint presentation running on a laptop through a handheld with wireless connection. Multibrowsing [10] is an example of dynamic UI distribution, as it enables users to move existing pages among multiple displays.

UI Migration

This dimension analyses whether there is some continuity when users change device and still access the same application. Users should be enabled to change the current device in use (the source device) and then have available the application on a different device (the target device) while the system automatically preserves the interaction state reached with the first device and offers an adapted UI on the new device. It is worth pointing out that distributed UIs and migratory UIs are two independent concepts: there may exist distributed UIs which are also able to migrate, but we can also have only distributed user interfaces (which do not migrate at all), or migratory UIs that are not distributed across multiple devices. In addition, multi-device UIs are different from distributed UIs: the latter is just an example, a particular case of the former. The range of values for this dimension is represented by the elements whose state can be preserved and transferred from one device to another: UI elements, functions, history, bookmarks, etc. However, it is worth noting that the state of the interactive application that can be captured/preserved depends on the implementation environment considered. For example, in Web applications it can include the state of forms, JavaScript variables, cookies, sessions, history, bookmarks, etc. An example of this is [8], which supports state persistence of web application in terms of state of HTML forms, session cookies, and JavaScript variables. Other types of applications (e.g. those supported by cloud computing) are able to preserve the state of only server-side information, others do not provide any support at all to UI continuity.

According to [17] continuity is considered to depend on how well the system supports cross-platform transitions, task migration and synchronization. An automatic solution for migrating UIs and preserving their state has recently been presented [2]. This approach, called Deep Shot, allows the migration of a user interface (or parts of it) by simply "shooting" it with a mobile phone camera. The authors claim that Deep Shot is compatible even with applications that are not Web-based. One limitation of this tool is that extra work is needed by developers to enable deep shooting/posting within an existing application.

UI Granularity

In this case we consider the granularity of the user interface that is manipulated (through e.g. distribution or migration) across various devices. As for the range of values we have:

- *entire UI*: the UI is seen as a single monolithic item, which can be e.g. moved/copied between devices;
- *groups of UI elements*: in this case we consider the possibility of e.g. distributing structured parts of user interfaces (e.g. navigation bars, articulated content areas with text and images, …) across various devices;
- *single UI elements*: in this case, single UI elements are distributed across devices;
- *components of UI elements*: interactive elements, which are usually characterised by *prompt*, *input*, and *feedback* are distributed across devices. For example, the user enters an input through a mobile device and the resulting feedback is shown on a large screen.

Some levels of granularity were addressed in [8], where a platform for totally/partially migrating Web pages across

devices is described. Various levels of granularity were also addressed in [11] to support UI distribution.

Trigger Activation Type

This dimension analyses how the request for a change in the cross-device user interface is triggered. This change could then activate e.g. a migration or a (re-)distribution of the UI. The simplest case is user-initiated: the user actively selects when, to which device and what should be changed. With automatic trigger the system autonomously activates the change when it recognises the verification of suitable contextual conditions (e.g. in case of a high battery consumption level and a simultaneous user's proximity to another device). Therefore, the system might decide that a device change is appropriate and then select the new device to be used. This type of automatic trigger can be related to the work on implicit human-computer interaction driven by the context discussed in [14], in which the system acts proactively on the basis of context information. Another option is a mixed type of trigger activation (partially suggested automatically and partially determined by the user): the system first automatically suggests a change to the user who is still able to modify some parameters in the request. In the case of the user-generated trigger we further distinguish between *push* and *pull* modalities depending on whether the triggered migration is from the local device to a remote one or vice versa. Therefore, the range of values is: *user* (which can be further decomposed into *push*/*pull* modality), *system*, and *mixed*.

In [8] both user (push and pull) and automatic migration triggering are available.

Device Sharing between Multiple Users

Multi-user interaction raises a wide variety of issues and possible solutions. Our framework is focused on support for multi-device environments. Thus, here we want to consider only the cases in which there are various devices and some of them can be shared by multiple users. This can happen either because the same device is targeted by the user interfaces of their applications (an example could be when two users use the same large display as a target for a migration from their mobile devices) or different users access the same interface on the same device (e.g. when two or more users exploit the same wall-sized interactive screen by using their own devices).

The multi-user/device framework discussed in [6] and the related scenarios explicitly consider the situation of several users concurrently accessing the same UI on a public display. Sharing implies that the supporting environment is able to indicate what the shareable devices are, whether there is any conflict in their use, and provide some information regarding their state. Thus, the possible levels of sharing considered are: multiple users can move information on that device (*sharing by moving*), or can even interact with that device (*sharing by interacting*).

Timing

Here the aspect considered is the time when a device change should occur in a multi-device configuration. An example typical case is a migration that has to be carried out as soon as the migration trigger is sent from the source device (*immediate* effect) in order to achieve seamless continuity. Another case covers the possibility for the user to specify the time when to defer the change in the multi-device configuration (*deferred* effect). This could be useful when the target device is temporarily unavailable to the user, hence the effect will be delayed until a more appropriate time. In this case the support should enable users to specify the device to be used as target, even if it could be temporarily unavailable in the current environment. The deferring time is implicitly managed in Deep Shot [2], which allows launching on a different device an application with a previously captured work whenever the user needs it. Myngle [15] could also be seen as an example of system that handles this possibility, since it lets the user choose when to restore the previous state of a Web application on a different device. Thus, the range of values for this dimension includes: *immediate*, *deferred, and mixed* (when both the previous ones are possible).

Interaction Modalities Involved

This dimension analyses the modalities involved in the multi-device UI. There are three possible values. *Mono-modality* means that the devices involved in the cross-device access support the same, (single), interaction modality. *Trans-modality* means that different devices can support different modalities, but any device supports only one modality at a given time. *Multi-modality* occurs when the multi-device interface simultaneously supports two or more interaction modalities in one (at least) of the devices involved. Various modalities have been considered in the dynamic interactors distribution proposed in [1], where the authors assume that each device category available in a smart environment has specific interaction resources (IRs). The approach is based on distribution of UI interactors among available IRs (devices). Distribution is performed automatically according to context information and developer/user settings. In their work, the authors refer to multi-modality as the combined use of multiple modalities within a (distributed) UI. However, they do not report on supporting more than one modality within the same device.

UI Generation Phase

This dimension specifies the phase when the user interface is obtained so as to be rendered on the target device(s). On the one hand, in the *design-time* case the UI is built in advance for each type of device, and then at run-time only the state has to be updated, in case of migration. On the other hand, the *run-time* case covers the situation where a run-time engine dynamically generates the user interface, according to the features of the target device. Also an intermediate approach (*mixed* case) is still possible where the supporting engine dynamically generates the user interfaces for the different devices by exploiting some

logical descriptions which have been created at design time. Therefore, we have three possible values.

The aspects related to the design time are discussed in [16], which presents Dygimes, a testbed for model-based UI development. Networked cooperating devices potentially offer a set of interaction resources to the mobile user. The authors distinguish between static and dynamic approaches for UI distribution, i.e. for distributing interaction resources (IRs) among UI components. The static approach would require knowing at development time the runtime context peculiarities, which is rather difficult since it implies to know which IRs are available in the environment and to which IR every part of a UI will be distributed. The dynamic approach allocates at runtime UI components to IRs, either automatically by the system or manually upon user request. The authors propose a possible solution describing the UI using a model-based approach. Different models are used to create descriptions at development time, while the actual UIs are generated at runtime starting from the models.

UI Adaptation Aspects

When changing the device(s) currently used, user interface adaptation is usually required. The adaptation process can have an impact at various granularity levels: either the entire application is changed depending on the new context, or just some logical UI parts (presentation, navigation, content) or even single UI components are adapted. There is also the case that no adaptation is provided, often generating low usability results.

By adaptation at the presentation level we mean, for example, the possibility to change the presentation layout. There are various ways to adapt the presentation ranging from simple scaling to applying information visualization techniques (e.g. semantic zooming, fisheye, …). Navigation refers to the connections among the different presentations: for example, when the number of presentations increases or decreases then the connections between them will be adapted accordingly. Content adaptation refers to when some information is removed, added, or modified (e.g. summarised) in order to produce a more usable UI depending on the resources of the device. One proposal [8] concerns a platform for partial migration of Web UIs with adaptation capabilities particularly useful when Web pages are migrated towards small devices: pictures are scaled, content can be split into several presentations, interaction components can be replaced by more suitable ones.

We identify three main approaches to adaptation: *Scaling*, in which the user interface is just linearly scaled according to the interaction resources of the available device, as it happens with Safari on IPhone; *Transducing*, an approach preserving the initial structure while translating the elements into other formats, and compressing/converting images to match device characteristics; *Transforming* goes further to modify both contents and structures originally designed for desktop systems to make them suitable to display on small screens. The multi-user/device framework previously cited [6] also tackles UI adaptation: transformation modules convert information into different representations (e.g. visual to audio) according to user's preferences. Transformations rely on previously defined annotations specifying how to convert resources at runtime.

Architecture

Two different strategies can be considered with regard to the architecture of a possible platform supporting a multi-device environment: *client/server*, in which there is an intelligent unit managing all requests and sending all data to target devices, thus controlling the user interface allocated across various devices; *peer-to-peer*, where the devices directly communicate and negotiate the distribution parameters. Within this type of coarse-grained distinction, it is still possible to identify more refined approaches for managing some phases, one of this being the interaction state preservation (when supported). For example, with a client-side approach we have some (client-side) mechanisms such as plug-ins or scripts that gather the interaction state data on the client and then send the collected data to the migration engine for further processing. With a server-side approach there is a server which is able to gather requests from the client side and consequently stores the relevant information. However, the choice between client/server and peer-to-peer may be driven by other aspects of the platform. For instance, if UI adaptation features are included in the platform, then the server support is highly desirable, since the computational effort needed to transform/generate the target interface might be too huge for a mobile device.

The migration platform described in [8] is based on a server which creates logical description of the source UI, adapts it to the target device capabilities and generates on-the-fly a specific implementation. The cross-device infrastructure of DeepShot [2] also relies on an instant messaging protocol which is server-based, even the devices involved in migration are usually co-located. On the contrary, a peer-to-peer strategy could offer more flexibility. For example, a set of devices equipped with peer-to-peer clients in the same network that do not rely on an external Web server could exchange information locally. This simplified architecture also leads to lower communication latencies between devices. An example of peer-to-peer implementation for distributed user interfaces is in [11], which requires the use of a specific development toolkit in order to benefit from this feature.

DISCUSSION and CONCLUSIONS

Table 1 provides a concrete example of how our logical framework can be used to analyse various proposals. For sake of brevity and lack of space we only consider a small set of tools.

The table shows that there are various points in the framework that are partially covered by most proposals and can be the topic for new research work. For example, there is a lack of solutions able to exploit peer-to-peer

communication among sets of devices that are opportunistically accessed. Another part that has received limited attention, also for its complexity, is the ability to preserve the state of UI functionalities in migration.

Aspect\Tool	Web Migration [8]	DeepShot [2]	Myngle [15]	Peer-to-peer DUIS [11]	Dygimes [16]	Multimodal distribution [1]
Distribution	Not Supported	Not Supported	Not Applicable	Dynamic	Dynamic	Dynamic
Migration	UI elements / functions	UI elements	Web history	UI elements	Not Applicable	UI elements
Granularity	Entire UI / Groups	Entire UI / Groups	Entire UI	Entire UI / Groups /	Entire UI / Groups /	Entire UI / Groups /
Trigger	User / Automatic	User	Automatic	User	Automatic	Automatic
Sharing	Move informat.	Not supported	Not supported	Move informat.	Not applicable	Not applicable
Timing	Immediate	Mixed	Mixed	Immediate	Immediate	Immediate
Modalities	Transmod.	Monomod.	Monomod.	Monomod.	Monomod.	Multimodal
Generation	Run-time	Mixed	Runtime	Runtime	Mixed	Mixed
Adaptation	Transduc./ Transf.	Scaling	Scaling	Transduc.	Transduc./ Transf.	Transducing/ Transfor.
Architecture	Client/Serv	Client/Serv.	Client/Serv	Peer-peer	Client/Serv.	Client/Serv.

Table 1: Example Application of the Logical Framework

Another topic that deserves further study regards users' attitudes towards multi-device UIs when the context is shared with other users, and while performing tasks with privacy concerns. This is a research direction that we also plan to further investigate in the future. In addition, the support of richer set of interaction modalities and their various combinations seem an area that needs to be better explored also considering the recent technological improvements for modalities such as voice and gesture.

To conclude, we can say that despite the increasing number of research proposals in the area of multi-device UIs, there is still need for further solutions able to address the parts of the logical space proposed that are still underexplored, such as the support for multimodal distributed user interfaces or peer-to-peer architectures for migration of Web applications.

ACKNOWLEDGMENTS

This work has been partly supported by the SMARCOS Project, http://www.smarcos-project.eu/

REFERENCES

1. Blumendorf, M., Roscher, D., and Albayrak, S. Dynamic User Interface Distribution for Flexible Multimodal Interaction, in *Proceedings of ICMI-MLMI'10*, Acm New York, 2010.
2. Chang, T.H., and Li, Y. Deep Shot: A Framework for Migrating Tasks Across Devices Using Mobile Phone Cameras, in *Proceedings of CHI 2011*, ACM Press, 2011, 2163-2172.
3. Coutaz J., Balme L., Alvaro, X., Calvary, G., Demeure, A., Sottet, J.: An MDE-SOA Approach to Support Plastic User Interfaces in Ambient Spaces. HCI (6) 2007: 63-72.
4. Dearman, D., and Pierce, J. It's on my other Computer!: Computing with Multiple Devices, in *Proceedings of CHI '08*, ACM Press, 2008, 767-776.
5. Demeure, A., Sottet, J.-S., Calvary, G., Coutaz, J., Ganneau, V., and Vanderdonckt, J. The 4C Reference Model for Distributed User Interfaces, in *Proceedings of ICAS '08*, IEEE, 2008, 61-69.
6. Ding, Y., and Huber, J. Designing multi-user multi-device systems: an architecture for multi-browsing applications, in *Proceedings of MUM* '08, ACM Press, 2008, 8-14.
7. Edwards W. K., Newman M. W., Sedivy J. Z., Smith T. F. Experiences with recombinant computing: Exploring ad hoc interoperability in evolving digital networks. ACM Trans. Comput.-Hum. Interact. 16(1): (2009)
8. Ghiani, G., Paternò, F., and Santoro, C. Push and Pull of Web User Interfaces in Multi-Device Environments, to appear in *Proceedings AVI 2012*, Capri, 2012, ACM.
9. Jimenez Pazmino, P., and Lyons, L. An exploratory study of input modalities for mobile devices used with museum exhibits, in *Proceedings of CHI 2011*, ACM Press, 2011, 895-904.
10. Johanson, B., Ponnekanti, S., Sengupta, C. and Fox, A. Multibrowsing: Moving Web Content across Multiple Displays, in *Proceedings of Ubicomp 2001*, Springer-Verlag, 2001, LNCS 2201, 346-353.
11. Melchior, J., Grolaux, D., Vanderdonckt, J., and Van Roy, P. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications, in *Proceedings of EICS 2009*, ACM, 2009, 69-78.
12. Myers, B.A. Using handhelds and PCs together, *Communications of the ACM*, 44 (11), 2001, 34-41.
13. Olsen, D.R., Nielsen, S.T., and Parslow, D. Join and Capture: a Model for Nomadic Interaction, in Proceedings of UIST '01, ACM, 2001, 131-140.
14. Schmidt, A. Implicit Human Computer Interaction Through Context. *Personal and Ubiquitous Computing* 4(2/3), 2000, 191-199.
15. Sohn, T., Li, F.C.Y., Battestini, A., Setlur, V., Mori, K., and Horii, H. Myngle: unifying and filtering web content for unplanned access between multiple personal devices, *Proceedings UbiComp* 2011, ACM, 257-266.
16. Vandervelpen, C., and Conix, K. Towards Model-Based Design Support for Distributed User Interfaces, in *Proceedings of NordiCHI 2004*, ACM, 2004, 61-70.
17. Wäljas, M., Segerståhl, K., Väänänen-Vainio-Mattila, K., and Oinas-Kukkonen, H. Cross-Platform Service User Experience: A Field Study and an Initial Framework, *Proceedings of MobileHCI 2010* (Lisboa, Portugal, September 2010), ACM Press 219-228.