

A Taxonomy for Migratory User Interfaces

Silvia Berti, Fabio Paternò, Carmen Santoro

ISTI-CNR, Via G. Moruzzi, 1
56100 Pisa, Italy
{Silvia.Berti, Fabio.Paterno, Carmen.Santoro}@isti.cnr.it
<http://giove.isti.cnr.it>

Abstract. Migratory user interfaces are particularly promising for forthcoming ubiquitous environments enabled by the evolution of wireless technology and the proliferation of a wide variety of interactive devices. In this paper we present a logical framework and some fundamental concepts and dimensions that can be useful to help user interface designers and developers understand migratory interfaces, analyse the state of the art, and identify areas which need further research. A number of works in this area are compared and referred to such framework and dimensions, so as to identify the advantages and drawbacks of the various approaches.

1 Introduction

Migratory interfaces are a fundamental aspect in forthcoming ubiquitous environments. Indeed, in the near future many users will access several types of devices in smart environments that allow them to actively and effortlessly switch among them (e.g., from a PC to a mobile phone), so as to furnish the best combination of application functionality and device mobility, and continue their work from where they left off. Therefore, migratory interfaces require session persistence and user interfaces able to adapt to the changing context (device, user, environment).

On the one hand, the effect of migratory interfaces is to make user interactions more natural and effective since they allow users to move about freely and still complete their tasks with the dynamic set of available devices. On the other hand, their design and implementation raises a number of non trivial issues that need to be carefully addressed. Thus, in order to aid understanding of the relevant problems to consider we have identified a reference framework composed of the logical phases characterising the migration process and a set of logical dimensions capturing the relevant aspects.

In recent years, research addressing migratory user interfaces has started in several centres. One early paper by Bharat and Cardelli [7] considered similar issues. They presented a programming model and an implementation of a tool for developing

migratory applications, placing no restriction on the kind of application that can be built. In this approach agents carry pieces of code and the state of the migratory application from one host to another, where a server allows the agent to rebuild the migrating application. Such an approach is unsuitable for supporting migratory interfaces, in particular in multi-device environments. In this context the goal is to support several types of platforms, from powerful stationary PCs to PDAs and cell phones. Most of them are mobile platforms, having to cope with power consumption issues, low storage and processing capabilities. The processing load involved with using agents that migrate to a platform hosting an agent server, where the application is rebuilt at runtime, would be too heavy for most of these platforms. Instead, in this paper we analyse, compare and contrast architectures and tools that tackle the challenges of user interface migration that can occur across heterogeneous devices. To this end, a taxonomy for migratory interfaces is proposed and discussed. It is worth pointing out that not all the approaches mentioned cover the whole migration process; often they just focus on a portion of it. In closing, we discuss the approaches considered and indicate areas that require further research in the near future.

2 Basic Concepts

Migratory interfaces are interfaces that can transfer among different devices, and thus allow the users to continue their tasks. This definition highlights important concepts: task performance continuity, device adaptation and interface usability.

The diversity in features of the devices involved in migration, such as different screen size, interaction capabilities, processing and power supply, can make a user interface developed for a desktop unsuitable for a PDA and vice versa. For example, an interface layout designed for a desktop platform does not fit in the smaller screen of a PDA, or a graphic interface running on a desktop system must be transformed to a voice interface when the application migrates to a car. Thus, an interface cannot migrate as is from one device to another (except in case of homogenous devices), and needs an intelligent engine in order to adapt it to the different features of the target platform taking into account usability principles.

By task performance continuity we mean that when migration occurs users do not have to restart the application on the new device, but they can continue their task from the same point where they left off, without having to re-enter the same data and go through the same long series of interactions to get to the presentation they were accessing on the previous device.

Moreover, the devices involved in the migration can belong to different platforms. The concept of platform groups sets of devices that share similar interaction resources (such as the graphical desktop, the graphical PDA, vocal platform). Interaction resources are atomic input and/or output channels (such as the screen, the microphone, the mouse). A device is a system able to run an application and support user interaction through a set of

interaction resources. In the migration process there are one or multiple source devices and one or multiple target devices. In total migration the interface migrates wholly from one device to another. In partial migration, only a part of the interface migrates to the target device. In distributing migration the interface migrates to multiple target devices, which is different from a distributed user interface, where the interface runs in one device and is allocated to multiple interaction resources connected to that device (for example two screens). In dynamic distributed user interfaces the allocation of the user interface parts to the interaction resources is dynamic (for example, moving one window from one screen to another or changing from graphical to vocal modality), but they are not migratory interfaces because the interface is always executed on the same device. In aggregating migration the user interface of multiple devices migrate into one device.

3 The Reference Framework

In the migration process two types of models are relevant (see Figure 1):

- *Context model*: refers to the description of aspects related to environment, user and device; for example, ambient noise, level of brightness; description of user profile and of the device features in terms of screen size, type of operating system etc.
- *Interactive system model*: refers to the description of user interfaces at different levels of abstraction, from the task model to the concrete interface description.

Figure 1 shows the steps in the migration process, where the ovals represent the activities and the rectangles the data manipulated and produced during the various phases. One key aspect in the migration process is the ability to capture the state of the migrating interface, which is the result of the history of user interactions with the application, including pages visited, data submitted and results of previous data processing. We suppose that there is a migration engine that collects the necessary information regarding the context, user interface, and requests for migration. In some settings (for instance, with peer-to-peer migration architectures), such activity is not allocated to one, centralised entity, but will be carried out by the different devices without the support of an external server.

At the beginning, the migration engine has to determine the basic parameters of the migration, namely the migration type and the target device(s) (see first top oval in Fig. 1). Then, it uses all this information in order to calculate the migration mapping and detect the interface state. This phase produces: the interface state, which is composed of all the user interface data resulting from the user interactions as well as the interaction resources and modalities associated with the target device(s); and the migration mapping, determining which activities will be supported by which devices as a result of the migration process.

In the adaptation performance step, the migration engine identifies how to adapt the user interface and its state to the target devices taking into account various aspects, such as user preferences. The result of this phase is the specification of the user interface

adaptation to perform, in terms of both interactors and navigation. Lastly, run-time support provides the resulting user interfaces in the target devices for the current state at migration time.

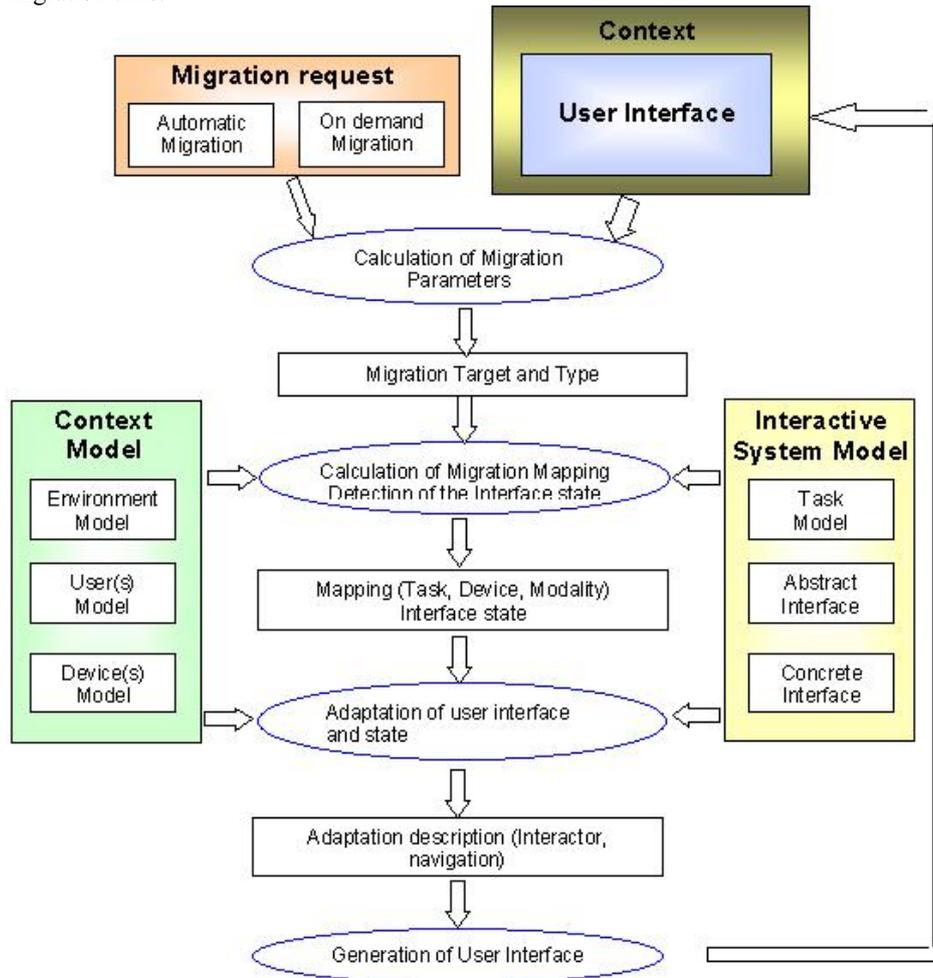


Fig. 1. The main steps and components of the migration process.

4 The Logical Dimensions in Migratory Interfaces

Several logical dimensions can be identified to indicate the aspects relevant to the migration process. They will be discussed in next sections.

4.1 Activation Type

This dimension analyses how the migration is triggered. The simplest case is *on demand*, in which the user actively selects when and how to migrate (the latter aspect generally involves also specifying the target migration device(s)). Otherwise, in *automatic migration*, it is the system that activates the switch. This can be done, for example, by checking some conditions, such as mobile device battery consumption level and device proximity in order to decide if migration is needed and to select the target device. In *automatic migration*, the target is selected automatically, for example considering the devices registered with the service according to their features (such as stationary or mobile, screen size, browser supported, etc), device usage (whether personal or suitable for shared use) and location (exploiting various tracking technologies). CamNote, an application complying with Cameleon-RT infrastructure described in [1], supports migration on demand when the user resizes the window and provides users with visual mechanisms aimed at making them aware of (the state of) the occurring migration (e.g.: users can see the application progressively disappear from one device and appear on another). Other approaches such as ISTI migration [5] support both on demand and automatic triggering. However, the vast majority of existing approaches supports only on demand migration. For instance, in the approach of ICrafter [15] the type of migration request considered is only on demand because the user is supposed to request a user interface for a specific service and for a specific device. The same type of activation is also supported in [16], where the user is allowed to save and restore multiple independent snapshots of web sessions on a browser and to retrieve them at a later time to continue any one of the sessions saved in any order.

4.2 Type of Migration

This dimension analyses the ‘extent’ of migration, as there are cases in which only a portion of the interactive application should be migrated. It is worth pointing out that a fine-grained migration (e.g.: partial) does not necessarily imply that a coarse-grained migration (e.g. total) is possible as well, since this dimension generally defines the scope of the possible migrations allowed by a certain approach. We identified five types of migration: *total*, *partial*, *distributing*, *aggregating*, *multiple*. Total migration basically allows the user to change the device used to interact with the application. In this case, the system is in charge of keeping interaction continuity and supporting interface adaptation to the different platforms.

Partial migration is the ability to migrate a portion of the user interface, while the remaining portion remains in the source device. In the *partial control migration*, there is a clear distinction of the portions (one for user interaction and one for information presentation) in which the migration occurs. An example of partial control migration obtained by analysing the logical description of the user interface is presented in [4]. Another example, in which the control part of one (or even more than one) service can

migrate to a single device can be found in the ICrafter approach [15], where the control part of different devices existing in an interactive workspace can migrate to a single device. If the client application is partially split into several parts (both concerning control and presentation), and such parts are distributed over target and source devices, we refer to *mixed partial* migration.

In the *distributing migration* the user interface is totally distributed over two or more devices after migration. This is different from distributed user interfaces (such as those considered in [17]) for which the user interfaces are originally generated as distributed among various interaction resources connected to the same device. The *aggregating migration* performs the inverse process: the interface of multiple source devices are grouped in the user interface of a single target device. The *multiple migration* occurs when both the source and the target of the migration process are multiple devices.

4.3 Number/Combinations of Migration Modalities

This dimension analyses the modalities involved in the migration process. *Mono-modality* means that the devices involved in the migration adopt the same modality interaction. *Trans-modality* means that the user can migrate changing the interface modality. An example of migration from graphical interface to vocal interface is the case of users navigating the Web through a PDA or Desktop PC and afterwards migrate the application to a mobile phone supporting only vocal interaction. Lastly, with *multi-modality* the migratory interface contemporaneously supports two or more interaction modalities at least in one device involved in the migration. Work in this area often has mainly focused on graphical interfaces, investigating how to change the graphical representations depending on the size of the screens available. For instance, both papers [8, 13] focus on how to adapt a user interface originally designed for a large screen to a small screen. In [5] there is an example of transmodal migration where conversion from graphical to vocal modality or vice-versa while migrating is obtained by exploiting task descriptions of the user interfaces and mapping the task support in a way appropriate to the interaction resources available.

Another work that started to consider the vocal modality and the possibility of generating voice interfaces is ICrafter [15], although still at a preliminary stage, since a lot of work must be done manually by the designer in order to provide suitable constructs for speech interactions.

4.4 Type of Interface Activated

This dimension specifies how the user interface is generated in order to be rendered on the target device(s). With *precomputed user interfaces* the user interface has been produced in advance for each type of device. Thus, at runtime there is no need for further adaptation to the device but only for adaptation of the state of the user interface. On the contrary, if a *runtime generation of user interfaces* is considered, the migration engine generates the

user interface according to the features of the target device at the time migration occurs. Between the two approaches, an *intermediate* one is still possible, in which the migration engine adapts dynamically to the different devices using some 'templates' or logical descriptions that have been previously created. The first version of the ISTI approach [3] used pre-computed user interfaces obtained through logical descriptions of the user interfaces generated for each platform through the TERESA tool. However, a more general solution has been designed and implemented in [2]: it starts with the desktop version of a Web application and when accesses from different platforms or requests of migration to different platforms are identified by a proxy server, then the underlying logical abstractions are rebuilt through a reverse engineering tool and they are used to dynamically generate the version of the user interface adapted for the target platform. In the approaches analysed, the most common style used is the intermediate one (see [1], [9], [10], [15]) which seeks a trade-off between the availability of pre-computed solutions and the ad-hoc support of the runtime generation.

4.5 Granularity of Adaptation

The adaptation process can be affected at various levels: the entire application can be changed depending on the new context or the user interface components (presentation, navigation, content). An example of adaptation at the *application* level occurs in the Aura approach [11]. In this case, suppliers provide the abstract services, which are implemented by just wrapping existing applications and services to conform to Aura APIs. For instance, Emacs, Word and NotePad can each be wrapped to become a supplier of text editing services. So, the different context is supported through a different application for the same goal (for example, text editing can be supported through MS Word or Emacs depending on the resources of the device at hand). By adaptation at the *presentation* level we mean, for example, when the presentation layout changes. *Navigation* refers to the connections among the different presentations: for example, when the number of presentations increases or decreases then the connections between them have to change. *Content* adaptation refers to when some information is removed, or added, or modified (e.g.: summarised) in order to produce a user interface more usable depending on the resources of the device. In *component* adaptation different representations of the same interaction object are supported. For example, the selection of elements can be obtained through list box, radio button, check box, etc. As for the layout and navigation adaptation, approaches such as [1], [8] and [9] support them. Furthermore, in the approach described in [6], the (Web) page is adapted and converted into a two level hierarchical organisation with a thumbnail page at the top level to provide a global view, and an index page to a set of sub-pages at the bottom level, so as to allow the user to select a particular region to zoom into for detailed information. Roam [9], CamNote [1], Dygimes [10] and ISTI [5] also operate at an interactor-based level by suppressing/replacing/transforming interactors. An example of interactor replacement is when a graphical link is replaced with a textual link.

It is worth pointing out that in [13] there is no change either in the content or in the layout of the original page.

4.6 How the UI is Adapted

There are several strategies regarding how to adapt user interfaces after a migration process occurs:

- *Conservation*: this strategy maintains the arrangement and the presentation of each object of the user interface: one possible example is the simple scaling of the user interface to different screen sizes.
- *Rearrangement*: in this case all the user interface objects are kept during the migration but they are rearranged according to some techniques (e.g.: using different layout strategies).
- *Increase*: when the user interface migrates from one device with limited resources to one offering more capabilities, the user interface might be improved accordingly, by providing users with more features.
- *Reduction*: this technique is the opposite of increase and it can be applied when the user interfaces migrates from desktop to mobile device because some activities that can be performed on the desktop might result unsuitable for mobile device support.
- *Simplification*: in this case all the user interface objects are kept during the migration but their representation is simplified, for example, different resolutions are used for figures or figures are substituted with textual descriptions.
- *Magnification*: this technique represents the opposite of simplification (e.g.: a textual description might be substituted with a multimedia information).

Examples of rearrangement can be found in [8], where the original Web page might be split into different sub-pages. It is worth pointing out that in techniques such as those supported in [6], it is the user who is in charge of interactively performing further adaptation, by collapsing irrelevant information so as to get more space for interesting data which are consequently magnified. Such an adaptation can also be saved for future uses.

4.7 The Impact of Migration on Tasks

The impact of migration on tasks depends on how the user interface is adapted because reduction and increase can produce some change on the range of tasks supported by each device. Differently, conservation and rearrangement do not produce any effect on the set of tasks supported. Then the possible cases are: 1) after a partial or distributing migration some tasks can be performed on two or more devices in the same manner (*task*

redundancy), which means, for instance, that the decomposition of the different tasks into subtasks is unchanged, as well as the temporal relationships (sequencing, concurrency, etc.) occurring among them; 2) after a partial or distributing migration a part of a task can be supported on one device and the other part/s is/are available on different devices (*task complementarity*). Additional cases are when the number of task supported 3) increases (*task increase*) or 4) decreases (*task decrease*) after migration. Obviously, a final case might be identified when the migration has no impact on tasks, as they remain substantially unchanged. The possibility that tasks increase/decrease has been taken into account in several approaches such as [9], [1], [5]. For instance, in Roam [9] there is a dedicated module (the *task manager*) whose job is to choose the appropriate tasks for the target device platform, and to remove widgets belonging to tasks inappropriate for the target device platform. For example, considering an e-shopping application, if UI designers specify that the *Add Shopping Item* task is not suitable for cell phones the corresponding widgets are removed by the task manager and not displayed on the cell phone presentation. A particular example of task increase has been identified in ICrafter, whose main novelty claimed by authors is to produce UI not only for services, but also for on-the-fly aggregation of services. Other approaches do not seem to pay particular attention to this dimension. For example, in the approach described in [13], the authors focus on five types of general, Web-oriented tasks such as finding information, re-finding information, etc., which are reasonably expected to be supported on most devices.

4.8 Context Model

During adaptation of the user interface the migration process can consider the context in terms of description of device, user and environment. Not surprisingly, the variable that is taken into account by all the approaches is the device and its properties. Sometimes the knowledge of the surrounding environment together with its characteristics has also been taken into account (see [15] and [1]) with the aim of producing more usable user interfaces. Information about the user is considered only in BSR. Indeed, one of the main contributions of BSR service is that it decouples association between browser state and a device, in favour of an association between browser state and the user. It introduces the concept of a *personal* repository where a user can store multiple snapshots and retrieve them anytime on any device. The benefits of this new association are that (1) it allows a user to switch devices in the middle of an active web session without losing state and having to restart on a new device, and (2) it allows a user to keep track of multiple active web sessions and freely save and continue any active Web sessions at any time from any device.

4.9 Implementation Environment

The migration process can involve different types of applications: Web-based (static/dynamic pages), Java, Microsoft etc. Probably due to their diffusion, the most recurrently considered applications are web-based applications ([5], [6], [8], [16], [12], [13]). The approaches that broaden their focus not only to web-based applications are CamNote [1], which is a slides viewer, ICrafter [15], which considers services in a workspace like projectors, scanners, etc., Roam [9], which considers seamless applications in general (defined as applications that can run on heterogeneous devices and migrate at runtime between heterogeneous devices) and [10], which focuses on .NET and Java applications. In addition, some requirements often need to be satisfied by the underlying software implementation, for example custom web plug-in, Java VM, etc. As this aspect is strongly dependent on the type of the approach used and the supposed goals, it might encompass a wide range of possibilities. For instance, the goals of Multibrowsing [12] are minimal configuration and robustness (dynamic discovery of displays in the vicinity enables the system to work transparently without workspace configuration even when target displays freely enter and leave the workspace). In addition, since it uses Web standards, multibrowsing accommodates any device or platform supported by the Web and leverages the vast existing body of Web content and services. The consequence is that in Multibrowsing a quite minimal pre-environment configuration is supposed to be carried out (a custom plug-in should be installed on clients so as to capture meaningful data). In other approaches the requirements might require little more effort. For instance, in ICrafter the applications, in order to be able to migrate have to extend the ICrafter APIs, while in Roam they have to support Java networking - based mechanisms like RMI and serialisation, since the system is based on the exchange of messages between the different components. Finally, approaches like BSR [16] put quite strong limitations to the kind of migration allowed as they only support migration between devices/browsers with similar capabilities (no multiplatform migration is allowed so far).

4.10 Architecture

With regard to the architecture of the migration service we can consider two different strategies: *client/server*, in which there is an intelligent unit managing all migration requests and sending all data to target devices; *peer to peer*, where the devices directly communicate and negotiate the migration parameters. Examples of client/server approaches are [5] and [15], in which the architecture is centred on a component called *Interface Manager* functioning as the intelligent engine of the system. On the other hand, an example of peer-to-peer approach can be found in Roam, where it is up to the Roam agents installed on both the source and the target devices to negotiate and decide in which terms the migration will be carried out. The architecture has an impact on how capturing

and saving the application execution state. First of all, we have a *client-side* approach where some mechanisms such as plug-in or applets collect the state on the client side and afterwards they send the entire data to the migration engine. Then, there is a *server-side* approach where a server detects each event on the client side and stores suitable information consequently. Lastly, we have a *mixed* approach, in which some mechanisms on the client side periodically captures the runtime data and sends them to the server. The most common approach is the client-based, used in [12] and [16]. For instance, in Browser State Repository Service [16] a client-based approach is used for capturing session state through a browser state snapshot.

4.11 Usability Issues

The adaptation and generation of user interfaces are performed following the main principles of usability such as web page layout consistency and support of appropriate tasks depending on different device. With respect to this dimension, we can say that all the approaches in which task adaptation to the device is carried out, might be considered attentive to the usability issue of selecting for the user only the tasks that are appropriate for each device. A more original aspect has been taken into account by [13], in which *transformation volatility* is considered. In this approach, the need for not changing too much the web pages in order not to change consequently the cognitive user's model is identified, so as not to disorientate/confuse the users that are already familiar with a certain web site.

5 Discussion and Conclusions

If we compare the different approaches with respect to the various dimensions that have been identified, it is possible to note that many approaches are unable to support automatic detection of context's changes and only address on-demand migration. In addition, the information from the context is often limited to that regarding the device(s). Moreover, Web-based applications are the most commonly considered applications in such approaches and only a few of them address a modality other than the graphical one (more than one approach simply focuses on moving from a large to a small-screen device). Furthermore, the problem of saving the application execution state, when addressed, is generally carried out on the client side. As for the adaptation, a number of techniques have been highlighted, and a great deal of attention has been paid especially to those techniques focussing on how to adapt pages originally targeted at desktop systems, so as to guarantee adaptation coverage for desktop-based applications. Lastly, the importance of considering how tasks might vary depending on the different contexts has been analysed, and various techniques to manage such variation highlighted (the most

common situation considered is the decrease in tasks due to a from-large-to-small device migration/adaptation).

In conclusion, we have presented a taxonomy for migratory interfaces, which allows designers to understand the relevant concepts and logical dimensions. In the discussion, we have also considered various approaches that have been proposed in this area. Our goal is also to highlight how the different issues raised by such interfaces have been addressed, in order to analyse advantages and disadvantages of the various methods. An analysis of how the considered approaches address the various dimensions can be useful to identify areas that are still problematic and require further research in the near future. For example, no approach has been able to address distributing migration involving multi-modal user interfaces or migrations where both the source and the target are composed of multiple devices, and there is still a lack of solutions able to support migration through peer-to-peer architectures.

Acknowledgments

This work has been partially supported by Vodafone and by the ISTI Curiosity-driven project MIGRANTES. We also thank colleagues from the previous EU IST CAMELEON project for useful discussions on the topics of this paper.

References

1. Balme, L., Demeure, A., Barralon, N., Coutaz, J., Calvary, G., CAMELEON-RT: a Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces, EUSAI 2004, pp. 291-302.
2. Bandelloni, R. Mori, G. Paternò, F., Dynamic Generation of Migratory Interfaces, Proceedings Mobile HCI 2005, Springer Verlag, Salzburg, September 2005.
3. Bandelloni, R., Paternò, F., Migratory User Interfaces able to Adapt to Various Interaction Platforms, International Journal of Human – Computer Studies, 60, pp. 621-639. Elsevier, 2004.
4. Bandelloni, R., Paternò, F., Flexible Interface Migration, Proceedings ACM IUI 2004, pp.148-157, ACM Press, Funchal, January 2004
5. Bandelloni, R., Berti, S., Paternò, F., “Mixed-Initiative, Trans-Modal Interface Migration”, Proceedings Mobile HCI 2004, Glasgow, September 2004, Lecture Notes Computer Science 3160, pp.216-227, Springer Verlag.
6. Baudisch, P., Xie, X., Wang, C., and Ma, W.-Y. Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content. Proceedings of UIST 2004 (technote), Santa Fee, MN, Nov 2004, pp. 91-94.
7. Bharat K. A. and Cardelli L., Migratory Applications. In proceedings of User Interface Software and Technology (UIST '95). Pittsburgh PA USA. November 15-17, 1995. pp. 133-142.
8. Chen, Y., Ma, W.-Y., Zhang, H.-J. Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices. WWW2003, May 20-24, 2003, Budapest, Hungary. ACM 1-58113-680-3/03/0005
9. Chu, H.-H., Song, H., Wong, C., Kurakake, S., Katagiri, M. Roam: a seamless application framework. The Journal of System and Software 69 (2004), pp 209-226.

10. Coninx, D., Luyten, K., Vandervelpen, C., Van den Bergh, J., and Creemers, B. Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems. MobileHCI 2003, Springer Verlag.
11. de Sousa, J., Garlan, D. Aura : an Architectural Framework for User Mobility in Ubiquitous Computing Environments. IEEE-IFIP Conf. 140 on Software Architecture, Montreal, 2002.
12. Johanson, B., Ponnekanti, S., Sengupta, C., Fox, A. Multibrowsing: Moving Web Content across Multiple Displays. Ubicomp2001, pp 346-353.
13. MacKay, B., Watters, C., Duffy, J. Web Page Transformation When Switching Devices. MobileHCI 2004, LNCS 3160, pp 228-239, Springer Verlag.
14. Nichols, J. Myers B. A., Higgins M., Hughes J., Harris T. K., Rosenfeld R., Pignol M.. "Generating remote control interfaces for complex appliances". Proceedings ACM UIST'02, pp.161-170.
15. Ponnekanti, S., Lee, B., Fox, A., Hanrahan, P., Winograd, T. ICrafter: A Service Framework for Ubiquitous Computing Environments. Ubicomp2001, pp 56-75.
16. Song, H., Chu, H.-H., Islam, N., Kurakake, S., Katagiri, M.. Browser State Repository Service. Pervasive Computing 2002, LNCS 2414, pp 253-266.
17. Vandervelpen, C., Coninx, K., Towards Model-based Design Support for Distributed User Interfaces, Proceedings NordiCHI 2004, October 2004, Tampere.