Platform Awareness in Dynamic Web User Interfaces Migration

Renata Bandelloni, Fabio Paternò

ISTI-CNR,Via G.Moruzzi 1, 56100 Pisa, Italy.
{r.bandelloni , f.paterno}@isti.cnr.it

Abstract. The goal of this work is the design of an environment for supporting run time migration of Web applications among different platforms. This allows users interacting with a Web application to change device and continue their interaction from the same point. The migration takes into account the runtime state of the interactive application and the different features of the devices involved. We consider Web applications developed through a multiple-level approach using: the definition of the tasks to support, the abstract description of the user interface, and the actual code. The runtime migration engine exploits information regarding the application runtime state and higher level information on the available target platforms. Runtime application data are used to achieve interaction continuity, while information on the different platform types involved are deployed to adapt the application's appearance and behaviour to the specific device.

1 Introduction

A wide variety of devices is now available on the market, and people are more and more likely to operate in a multiplatform environment where different platforms have different interaction capabilities. Many efforts are currently aimed at allowing users to interact through multiple devices. A framework discussing the issues associated with applications that can be spread over different surface areas, each supporting diverse user interaction techniques, is discussed in [1]. In our work we focus on Web-enabled platforms. For example, a user browsing the net with a PDA touch screen or a mobile phone keypad would be more comfortable using the mouse and keyboard of a stationary PC. Conversely, a user may be entering private data through a stationary PC and wish for the greater privacy afforded by a PDA. In both cases, a multiplatform migration service would be necessary, by which the user could interact with web applications while changing devices and still maintaining interaction continuity. There are two main issues concerning this kind of service. Firstly, the diversity in features of the platforms involved in migration, like different screen size, interaction facilities, processing power and energy supply, can make a Web application developed for a desktop, unsuitable for a PDA and vice versa. Thus, an application cannot migrate as it is from one device to another, and must be adapted at runtime, taking into account the diversity of the devices involved [2]. The second issue concerns

interaction continuity. Users who want the application to migrate, do not want to have to restart the application on the new device; they want to continue their interaction from the same point where they left off, without having to re-enter the same data and going through the same long series of links to get to the page they were visiting on the previous device [3]. Two main kinds of information are relevant in performing migration: static information refers to the features of the devices, whereas runtime information refers to the state of the migrating application that can be summarised by the history of user interactions with the application, including visited pages, submitted data and results of previous data processing. There are several techniques for migrating user interfaces to different devices, in particular to small screens, and most of them rely on size reduction and data summarisation [4], with the risk of making the application unusable because objects on the page are difficult to recognise. Herein we focus on interaction continuity and device adaptation at runtime that takes into account usability principles. We consider different platform-specific versions of the same application, starting with a general task model [5] from which we generate the actual application by means of the TERESA tool [6]. We take into account the migration of TERESA-generated applications, for which a description of the pages and the interactions that they support, are produced by the tool itself, at different abstraction levels. Runtime data on the state of the application for which migration is required will be collected locally from the platform requesting migration. This information is transmitted to the server in order to recreate the corresponding state in the application for the target device.

2 Generating Device Aware Web Applications.

We consider Web applications developed through a multiple-level approach able to obtain versions suitable for different kinds of devices and platforms. The starting point is the task model of a nomadic application that can be accessed through different platforms. The general model is refined for each of the specific platform that must be supported by the application and by means of the TERESA tool, different implementations are generated fitting different platform features and according to usability principles. The main levels involved in the generation process are:

1. *Task Model (TM)*: describes the logical activities that must be performed by users in order to reach their goals. A set of attributes is defined for each task and tasks are composed by semantic and temporal relations.
2. *Abstract User Interface (AUI)*: defines the main characteristics of the interaction objects supporting task performance, abstracting from low level details. AUIs are defined in terms of presentations identifying the set of user interface elements perceivable at the same time. These elements are represented as interactors being Abstract Interaction Objects (AIO) described in terms of their main semantic effects.
3. Concrete User Interface (CUI): this is the implementation level, the actual user interface produced for a specific device in a given implementation language as Java, XHTML and so on.

3 Runtime Migration Cases.

Different types of runtime migration can be identified, along with different levels of complexity for each one of them:

- *Total Migration*: the client application migrates totally from a device to the other.
- *Control Migration*: the client application is divided into two parts, one for user interaction (control part) and one for information presentation (presentation part). The control part remains on one device, while the presentation one migrates to the other device, or vice versa [7].
- *Mixed Migration*: the client application is split into several parts, concerning both control and presentation and different parts are distributed over two or more devices.

In our work, we focus on *Total Migration*, with the goal to support a runtime migration that takes into account the differences between the two platforms involved. TERESA structures an interactive application into presentations and transitions among them. When we migrate a presentation from a platform to another one the runtime support first identifies the closest presentation in the target platform. The difference between presentations in different platforms is calculated in terms of the number of logical tasks supported. A task can be supported through different interaction techniques. However, the logical meaning of the task is still the same. Taking into account interactive applications developed by means of TERESA we can identify the following situations concerning the runtime migration of a presentations between two platforms:

- The migrating presentation corresponds to one target presentation supporting:
    - Same number of tasks.
    - Lower number of tasks.
    - Higher number of tasks
- The migrating presentation corresponds to multiple target presentations that make an exact partition of the task set associated with it.
- Multiple presentations in the source platform correspond to one presentation in the target platform.

4 Our Migration Solution

Information concerning the platform asking for migration, and the state of the application running over it, is collected and elaborated in order to activate the application on the target platform without losing interaction continuity. Since the presentation number and the tasks supported by the various platforms can be different, it is not possible to create a one-to-one correspondence between presentations for different platforms. Source and target platform versions are generated by TERESA separately, using the information contained in the two corresponding task models. One important issue is how to identify the presentation for the target platform corresponding to the

one active on the platform requesting migration while maintaining the state of its interaction objects. The run-time state, consisting of the visualized page and the state of its objects, is mapped first onto the corresponding abstract presentation and then onto the corresponding set of tasks. The page to be visualized on the target device will be identified using the inverse process: from the set of tasks to support the tool identifies the most similar abstract presentation and then the corresponding page in the application version for the target platform. Similarity is calculated in terms of tasks supported, the more the tasks associated to the two presentations are similar, the more the presentations are similar. Presentation similarity is the basic criterion to be considered, but under particular conditions it cannot be enough. When the migrating presentation supports a task set that is associated with multiple presentations in the target version, each of them supporting the same number of tasks, similarity degree will be the same for each potential target presentation. Thus, a further criterion should be used to decide which target presentation to activate. To this end, we use the identification of the target presentation supporting the task associated with the interaction object last modified by the user, since it is more bound to continue interaction from that point.
Once the corresponding presentation has been identified, it is necessary to calculate the state of the objects contained in the page that will be sent to the target device. For this purpose, runtime data referring to the runtime state of the application will be associated to the corresponding AIOs and adapted to the object implementation for the target device.

5 Migration Service Architecture

We aim at supporting Web application migration for a wide variety of devices like desktops, laptops, PDAs, cellular phones and generally any device able to access the Internet through a browser. Our migration service relies on a server machine working both as a Web server storing the platform specific implementations of the application, making them accessible to client platforms, as well as a migration server, managing context information to support migration requests. Client platforms use the migration client loaded from the server in order to enabling or disabling the possibility of receiving incoming applications and migrating Web applications. References to all platforms, which enabled the reception of incoming applications, are stored in the server. When a platform asks for migration, the request sent by the locally running migration client, reaches the migration server, which will deploy both runtime and static context data to perform the presentation mapping process as described in Section 4. The corresponding page and its runtime context for the target device will be finally sent to the migration client that will open locally a browser window allowing the user to continue its interaction (the sequence of functionalities to perform is indicated in Figure 1).

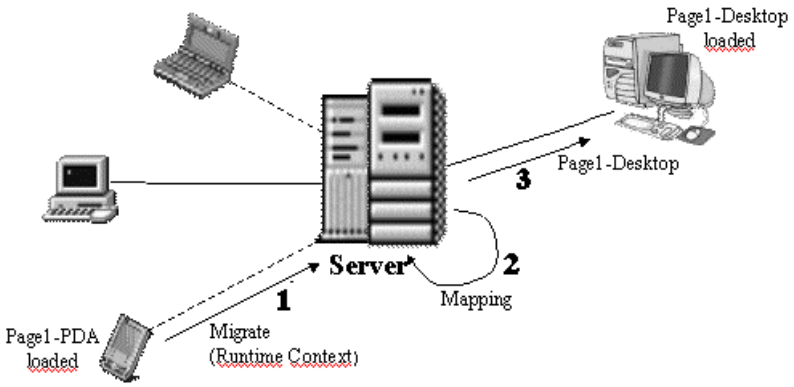

Figure 1. The Migration Process.

6 Conclusions and future work.

We have discussed an architecture to support migratory Web interfaces. A first prototype for total migration of applications obtained through the TERESA tool has been developed. We are now improving the collection of run-time state data in order to make it more complete and improve the support of interaction continuity. Future work will be dedicated to extending this approach in order to address other types of user interface migrations.

This work has been supported by the CAMELEON project (http://giove.cnuce.cnr.it/cameleon.html). We thank colleagues for useful discussions.

References

1. J.Coutaz, C. Lachenal, S. Dupuy-Chessa. *Ontology for Multisurface Interaction*. Proceedings INTERACT 2003. IOS Press. Zurich, September 2003.
2. A. Kaikkonen and V.Roto. *Navigating in a Mobile XHTML application*. In Proceedings of CHI 2003. Ft. Lauderdale, Florida, April 5-10, 2003. Vol.5, pp. 329-336.
3. H. Song, H. Chu, S. Kurakake. *Browser Session Preservation and Migration*. In Poster Session of WWW 2002, Hawai, USA. 7-11. May, 2002. pp. 2.
4. B. MacKey. *The gateway: A Navigation Technique for Migrating to Small Screens.* Doctoral Consortium, CHI 2003. Ft. Lauderdale, Florida, April 5-10, 2003. pp. 684-685.
5. F.Paternò, C.Santoro, *A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms, Interacting with Computers*, Vol.15, N.3, pp 347-364, Elsevier, 2003.
6. G. Mori, F. Paternò, and C. Santoro. *Tool support for designing nomadic applications*. In Proceedings of IUI 2003 . ACM Press, 2003. pp. 141–148.
7. J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, M. Pignol. *Generating remote control interfaces for complex appliances*. Proceedings ACM UIST'02. October 27 – 30. Paris, France. Vol.4, pp.161-170.