Authoring Multi-device Web Applications with Database Access

Giulio Mori, Fabio Paternò, and Carmen Santoro

ISTI-CNR, Via Moruzzi 1,
56126 Pisa, Italy
{Giulio.Mori, Fabio.Paterno, Carmen.Santoro}@isti.cnr.it

Abstract. In this paper we present an environment for authoring Web sites through a model-based approach for user interface design. In particular, we focus on how it supports the access to remote databases and the dynamic generation of the Web pages presenting the corresponding query results. The environment is able to support development of applications implemented in many Web mark-up languages (XHTML, XHTML MP, X+V, VoiceXML) adapted to various interaction platforms (vocal, mobile, desktop, …).

Keywords: Dynamic Web Applications, Multi-device Environments, Interfaces Adapting to the Platform.

1 Introduction

The increasing availability of interaction devices poses a number of challenges to Web designers and developers to obtain usable interactive applications. There is an increasing number of Web applications that would benefit from the possibility of accessing them through many interaction platforms that can even vary in terms of the modalities supported (graphical, vocal, gestural, …). To address such issues the W3C consortium started various activities in the area of Ubiquitous Applications [3] and Multimodal Interfaces [5] and a number of mark-up languages have been developed, including XHTML, XHTML Mobile Profile (MP), VoiceXML, X+V.

In this context, model-based approaches have shown to be useful because they provide logical descriptions independent of the implementation languages and allow designers to better manage such increasing complexity. Many types of model-based approaches have been proposed: in the software engineering area there are several tools based on UML; for data-intensive Web applications there are approaches such as WebML [2]; examples of model-based approaches in the user interface area are UsiXML [7] and TERESA XML[4]. In the latter community, there is a general consensus regarding the useful logical descriptions [1][6] supporting user interface design: the *task and object level*, which reflects the user view of the interactive system in terms of logical activities and objects manipulated; the *abstract user interface*, which provides a modality-independent description of the user interface; the *concrete user interface*, which provides a modality-dependent but implementation language-independent description of the user interface; The *final implementation*, in an

L. Baresi, P. Fraternali, and G.-J. Houben (Eds.): ICWE 2007, LNCS 4607, pp. 182 – 187, 2007.

implementation language for UIs. In particular, when designing multi-device user interfaces, this framework has the additional advantage that the task and the abstract level can be described through device-independent languages: it means that it is possible to use the same languages for whatever platform we aim to address.

In this paper, we present how an approach for model-based user interface design is able to support access to remote databases and the dynamic generation of the Web pages presenting the corresponding query. We discuss the solution proposed at both conceptual and implementation level. An example application is also presented, followed by empirical feedback collected from some Web designers and developers.

2 Design of Multi-device Applications with Database Access

Our approach is based on a tool, TERESA [4], which applies transformations and logical descriptions to generate interactive applications for different devices starting with different abstract models. Here we present an extension of the approach able to overcome the two main limitations of the original environment: generation of only static Web pages, and need for providing designers with greater control over the UI generated. In order to enhance the environment with features able to support access to remote databases, we provided support through different abstraction levels and suitable transformations among the different levels indicated in the introduction, including the generation of the appropriate code of the final user interfaces for various target platforms starting from specifications at the concrete level.

As for the final implementation language considered, for this type of support we focus on JSP, a well-known language for generating dynamic web pages. Therefore, if the application built by the designer contains at the concrete level objects supporting access to a remote database, the tool automatically produces dynamic JSP pages to provide support to this feature, which then will generate the implementation languages depending on the target platform. In the case of generation of applications for database access, the tool also generates some servlets that have to be installed in the server side in order to make the environment working properly. On the server side, the servlet is in charge of receiving the query specification, connect to the concerned database and execute the query to the database, sending the result of such query to a presentation. The environment for the generation of pages accessing to remote databases can provide different implementation languages (such as XHTML, XHTML Mobile Profile, VXML…) for the modalities supported by the platforms. We can analyse the new support by considering the possible abstraction languages.

2.1 The Task Level

The task models are specified in the ConcurTaskTrees notation [6]. At the task level, the allocation of tasks (namely, if they are performed by the application, or the user, or an interaction between the user and the system) and the objects manipulated by the tasks are important information for modelling and supporting the access to a database. Indeed, if we consider the specification of a task requiring access to a remote database, we should include an interaction task through which the user defines the attribute values; then, another (application) task is supposed to be sequentially connected to

the first one (from which it receives input values) and able to send such values to the back-end module of the server accessing the database. The last task is another application task, providing the user with the presentation of the query result. As for the specification of the objects manipulated by the tasks, they are classified in *perceivable* and *application* objects. The former have a direct impact on the UI as they are associated with concrete interface elements (menus, buttons, labels, etc.), whereas the "application" objects refer to logical objects corresponding to elements of the underlying application. When the task model is transformed into an abstract interface specification, tasks corresponding to database access are transformed into abstract interactors (called *activators*), which are associated with the functionality of the core that should be accessed and other attributes indicating the request parameters. The tasks presenting information to the user can be automatically detected because they are application tasks manipulating both perceivable and application objects. We will see that they will be supported by a new type of object at the abstract user interface level, the *table*, mapped at the concrete user interface level onto a *database_table* object.

2.2 The Abstract and the Concrete User Interface Description level

In TERESA XML, the logical interfaces are structured into presentations. To support access to remote database, at least two presentations are needed: one for allowing the user to specify the values of the query parameters and to send them to the associated database; another one for showing the query results. In terms of Abstract Interface, the first presentation is composed of a group of interactors allowing users to edit the query, which are related to an object of *activator* type (supporting the triggering of the functional module sending the values to the database). The second presentation includes the elements that will contain the query result. This type of element at the abstract level is a *description*, mapped at the concrete level onto a new type of element, a *database_table*. For the first presentation, the input elements, such as the *textedit* objects are mapped into the concrete elements of type *textfield*, whereas the *activator* is mapped onto the *activate_database* concrete element, introduced to support this new feature. As we said, the object *activate_database*, which has been introduced in TERESA XML as a new type of *activator* interactor, enables the connection with the concerned database. It has a number of attributes for specifying the parameters necessary to build the database query, e.g. *label* (for naming the concrete object), *database_properties* (for specifying the properties of the considered database, e.g. the server on which the database resides, the name of the database, user account), *attributes_names* (the set of attributes on which queries will be possible), *presentation* (the name of the presentation that will visualise the query result). Figure 1 shows how the *activate_database* element is supported within the TERESA authoring environment: in the left part there is the list of edited presentations, in the right top part there is the abstract specification of the currently selected presentation (the presentation titled "Ford News"), in the right bottom part there is the concrete description of the currently selected element of the abstract part (*Activator_7_0*). In the concrete user interface language for graphical interfaces we have also introduced the *table* element for managing the data corresponding to the query result. In particular, two types of tables were introduced: *database_table* and *normal_table*. The first one is used when the content of the table is not known at design time, as it is filled at runtime with the

query result. Thus, the designers do not know the number of its rows, but they should know the number of columns (database attributes). The *normal_table* is a table whose content (both textual and graphical content) is statically decided by the designer.

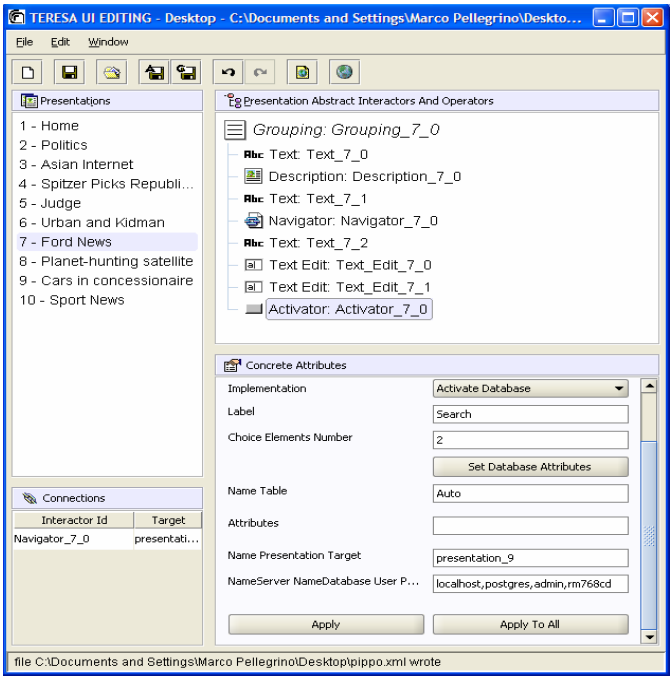

Fig. 1. The specification of the activate_database element with the TERESA tool

For both tables it is possible, with the tool, to customise their visualisation (modifying parameters such as colour and size of the table border, background colour, etc.).

3 An Example Application and a First Usability Evaluation

In this section we analyse an example application, working for both the desktop and the mobile platform, which provides the user with a list of up-to-date news. The user can query a database to get more information about a specific topic (eg: cars). Fig.1 shows the TERESA environment while editing the presentation allowing users to specify the query to the concerned database about cars. In this case, the attributes include brand name, type of fuel, number of car doors, etc. As the desktop is supposed to have good capabilities in display size, the designer chooses to visualise all the attributes: indeed, no attribute is specified in the related panel field labelled ”Attributes”. For the mobile platform, the number of attributes to be visualised should be more limited, and listed by the designer in the field “Attributes”.

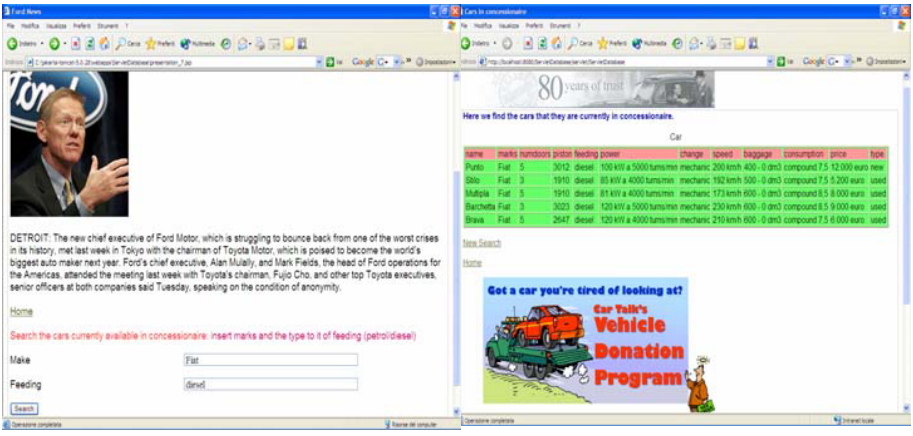

Fig. 2. The final user interface for the considered example (desktop platform)

In Figure 2 shows the user interface to specify the query (left) and the corresponding information result regarding all 12 attributes. For the mobile platform the case is different. Not only the designer has to select a more limited number of attributes to be visualised in the result (namely, the columns of the table), but, we have also to limit the information to be shown in each presentation (namely, the number of rows that can be reasonably presented in each table), because of the limited size of the mobile platform screen. In the example we decided to present only two attributes (car model name, and power type), and to enable the visualisation of only five rows in each presentation. The result of such settings produces the presentations shown in Fig. 3. As you can see the tool automatically adds the links required to navigate through the various pages generated for presenting the query result.

Fig. 3. The final user interface for the considered example (mobile platform)

A first evaluation session was performed to assess whether the new version provides designers of dynamic user interfaces with useful support and more control on the generated UIs, and its usability. A first test was carried out involving 5 developers recruited from the institute community, ageing between 25 and 38, and with laurea degree in Informatics. Before the exercise, users read a short introduction text about the tool and then instructed about the task that they were expected to carry out: building an application able to access a database with the support of the tool. Differences initially noticed between people having some knowledge of the tool and people who had not soon disappeared as soon as users gained familiarity with it. The intuitiveness of the tool was rated good (the average value was 3.5 in a (min) 1-to-5 (max) scale), although improvable. The pages built with the tool were judged usable (average rating: 4); testers reported that the final UI reflected their objectives, showing that the tool provides a good control on the UI produced (average rating: 4). Almost all users judged extremely valuable the help provided by the tool to the designers during the building of UIs accessing to remote databases (average rating: 4). Especially useful was considered the flexibility given by the tool in combining such dynamic objects with more static parts of the user interface.

4 Conclusions and Future Work

In this paper we present a new tool for supporting generation of interactive Web applications for various types of devices and able to access remote databases. The solution developed is able to support authoring of applications for desktop and mobile platforms, and generate page implementations in XHTML, XTML MP, VoiceXML (only vocal interaction) and X+V (vocal and graphical). Future work will be dedicated to further testing it in order to receive empirical feedback regarding its usability and suggestions for further improvements.

Acknowledgments. We thank Marco Pellegrino for the help in the implementation.

References

1. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.A: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15(3), 289–308 (2003)
2. Ceri, S., Fraternali, P., Matera, M.: IEEE Internet Computing, 6, 4, July/August, pp.20–30 (2002)
3. Ubiquituous Web Application Activity: http://www.w3.org/2007/uwa/
4. Mori, G., Paternò, F., Santoro, C.: Design and development of multidevice user interfaces through multiple logical descriptions. IEEE Transactions on Software Engineering 30(8), 507–520 (2004)
5. Multimodal Interaction Activity, W3C, http://www.w3.org/2002/mmi/
6. Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Springer, Berlin (1999)
7. Vanderdonckt, J.: A MDA-compliant environment for developing user interfaces of information systems. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 16–31. Springer, Heidelberg (2005)