DESIGNING AND DEVELOPING MULTI-USER, MULTI-DEVICE WEB INTERFACES

Fabio Paternò, Ines Santos
ISTI-CNR, Pisa (Italy)

Abstract: The need for support of multi-user interaction is growing in several application domains, including the Web. However, there is a lack of tools able to support designers and developers of multi-user, multi-device interactive applications. In this paper we present a proposal for this purpose describing how it can provide support at both design and run-time. The design and development process can start with task model descriptions and such logical information is used to generate interfaces adapted to the target platforms and mechanisms for their coordination at run-time.

Key words: Multi-User, Multi-Device User Interfaces, Authoring environments, Model-based design, Software architectures.

1. INTRODUCTION

The Web is the most common user interface. In this area there is an increasing number of applications that can be accessed through both desktop and mobile devices. Given the progress made in network technology and access, we have reached a stage in which team-work is widely supported even in Web applications (see for example [1]). Unfortunately, in most systems data is organized and published on the Web using "Web oriented database systems". This type of application permits only asynchronous work. Some support for cooperative applications is provided by workflow systems, which are able to coordinate the tasks performed by various users (see for example [2]) but they are not able to support synchronous work (multi-user applications that allow cooperation between several users in real-time). The few existing authoring environments for synchronous work applications do

not take into account that users can access them though different types of devices.

The goal of this paper is to present an environment supporting the design and generation of Web synchronous applications (same time, different place) and an associated architecture able to support them at run-time. Our environment allows developers of collaborative applications to focus on the tasks that users wish to accomplish. The importance of such task-oriented approach for cooperative application has been shown in several works, see for example [3][4]. To this end, it allows them to identify, implement and iterate over the main aspects of the cooperative applications. Since tasks in some applications must be performed by a user before another user can begin a different task, the desired order of task performance must be preserved in the resulting cooperative system. Thus, the run-time architecture must have a knowledge of the task model and how it is mapped onto the user interface implementation in order to make them consistent in terms of the possible dynamic evolution. While tools supporting model-based approaches to user interface design and development exist, see for example [5][6][7], none of them has addressed the possibility of supporting generation and run-time support of multi-user, multi-device synchronous applications.

In the following we first discuss background and related work. Then, we show how cooperative multi-user applications can be designed and how the corresponding user interface implementations adapted to different platforms are obtained. Then, we present the corresponding software architecture, including mechanisms for synchronous cooperation. We also briefly describe an example application. Lastly, we draw some conclusions and provide indications for future work.

2. BACKGROUND AND RELATED WORK

The new environment presented in this paper has been obtained by extensively redesigning a previous tool (TERESA) [6]. TERESA is a multi-device, single-user interface authoring environment that was developed to allow designers to start with a task model in order to obtain different user interfaces for different target platforms. By platform we mean a set of devices that share similar interaction resources (such as the graphical desktop, the graphical mobile device, the vocal device). In the development process, logical descriptions of the user interface are used as well: the abstract interface description is a modality-independent description and the concrete interface description is a modality-dependent refinement of the abstract one; both are implementation-language independent. Thus, for

example at the abstract level we can say that we need a selection object without any indication of how the selection can be performed: it could be a graphical selection or a vocal command or a gesture. At the concrete level, we assume a modality and we indicate an interaction techniques supported by such a modality. In the case of a graphical desktop platform, we can indicate a list or a radio-button or a pull-down menu to perform a selection. In the logical user interface descriptions there are logical interactors (such as edit, select, activator, description) and logical composition operators indicating how they should be put together (such as grouping, hierarchy, ordering). Thus, depending on the target platform, when designers move from the abstract to the concrete level, different sets of possible interaction techniques are considered, which are then implemented according to the target implementation language (that can be, for example, XHTML, Java, C++). However, TERESA supported only single user interfaces. Our new tool extends its functionality in many respects in order to support the design and development of Web applications involving cooperation among multiple users performing different tasks through different devices.

Some environments for development of multi-user interfaces already exist: for example, GroupKit [8] is a toolkit that simplifies the development of groupware applications to support distance-separated collaborative work. However, it does not address the issue of supporting interaction among multiple users interacting through different device types. WebSplitter [9] aims at supporting collaborative Web browsing by creating personalized partial views of the same Web page depending on the user and the device. To this end, developers have to specify the Web content in XML and define a policy file indicating the tags content that can be presented depending on the user and the device. In addition, they have to define XSL modules in order to transform the XML content into HTML or WML. At run-time a proxy-server generates the Web pages for multiple users, which provide each user with a presentation depending on his/her privilege and device. This approach has several drawbacks. Developers have to manage a plethora of low-level details to specify XML content, policy files, and XSL transformations.

We have similar goals but have adopted a different solution using logical descriptions of interactive systems. They are still specified using XML-based languages but developers can work directly on the logical representations without having to learn the many possible implementation languages. As Greenberg has indicated [10], we aim to remove low-level implementation burdens and supply appropriate building blocks in order to give people a language to think about their user interface and allow them to concentrate on creative design. In addition, we are able to support the design

of collaborative applications, in which the results of the actions of one user can change the interface of another.

Other work in this area is the thin-client groupware [11]. It focuses on the use of server-side software components to obtain low-resource collaborative client solutions (chat, email, etc.) through some basic mechanisms provided for this purpose. Our solution differs in that it is based on the use of logical descriptions that provide designers with easier to manage representations of the underlying implementation.

3. THE AUTHORING OF MULTI-USER INTERFACES

Our environment allows designers to develop multi-user, multi-device interfaces starting with the task model of the application. It can be graphically specified in the ConcurTaskTrees notation, a hierarchical notation providing the possibility of specifying temporal relations among tasks as well as the objects they manipulate and a number of their attributes. The task model of a cooperative application is developed in such a way that the task model of each role involved can be specified separately. In addition, designers can specify the cooperative part. The purpose of this part is to indicate temporal and semantic relations among tasks performed by different users. Thus, it indicates high-level cooperative tasks, which are activities that are decomposed into subtasks performed by different users, and specifies their refinement, down to the corresponding basic tasks performed by each user, along with their temporal constraints. Then, the tool is able to firstly transform the task model into a logical user interface description and then into a user interface implementation. Before starting such transformations designers have to indicate the target platform for the user interface of each role. Thus, the tool supports iterative refinement and generation of interfaces structured according to the task model structure and selects the implementation techniques depending on the interaction resources available in the target platform. For example, a high cardinality selection can be implemented with a list in the desktop and with a pull-down menu on a mobile device in order to consume less screen space.

Figure 1 shows the authoring environment when the logical interface is obtained from the task model. The logical user interface is structured into presentations listed in the top-left area. In the top-right part, the abstract description of the selected presentation is highlighted, whereas the bottom-right displays the possible refinement of the currently selected element in the abstract part for the current target platform.

Another addition in this new environment is the possibility of generating interaction techniques supporting communication among users. Thus, for example, while in the single user environment it is possible to associate the activator interactor with calls to content-server or local functions, in this new version it is also possible to include a type of activator interaction, which sends information to the user interface of another user. In this case, the designer should also indicate the user role and the interactor identifier in the corresponding interface that is to receive the transmitted information.

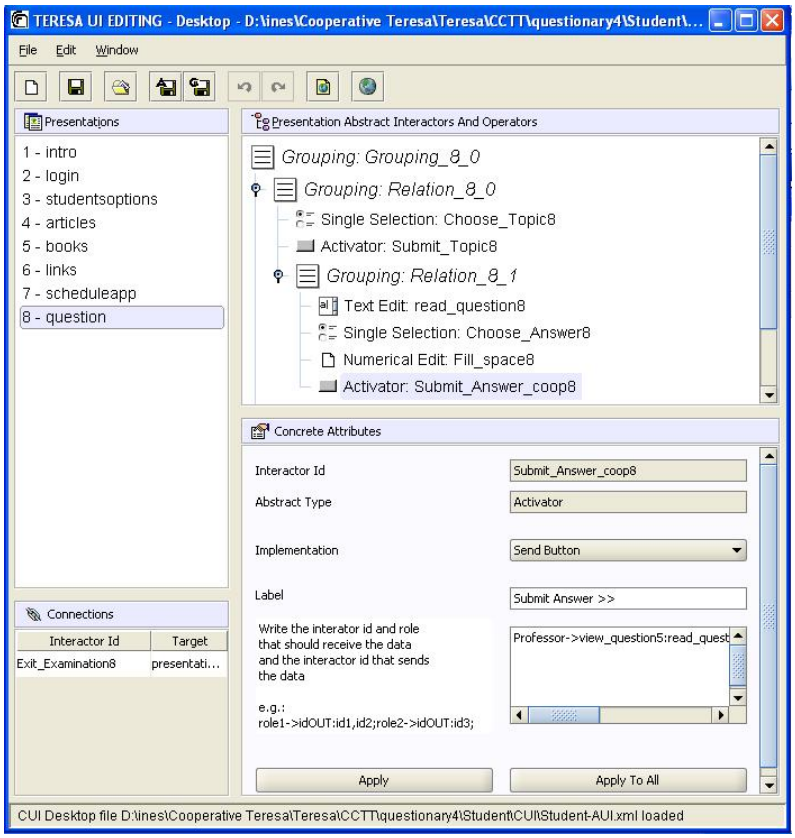

Figure 1. The Cooperative TERESA authoring environment.

For example in Figure 1, the lower section of the part associated with the concrete attributes provides a text box where the designer specifies that the information corresponding to the value of the view question5 interactor should be sent to the Professor when this button is selected and should also be associated to the value of the read_question8 interactor of the corresponding user interface. More in general, it is possible to indicate the

value of one or multiple interactors of the current user interface and specify one or multiple interactors in the user interfaces of the target roles that should receive such values.

For each role, once the corresponding concrete interface has been defined with the support of our authoring environment, then the system generates the corresponding user interface for the indicated platform. For example, Cooperative Teresa generates XHTML implementations for desktop platforms and XHTML Mobile Profile implementations for mobile platforms. This last transformation is performed in a modular way so as to account for both the general settings and the indications provided by the concrete interface regarding the techniques used to implement both composition operators and single interaction objects. Thus, for the implementation of interaction elements grouping, for example, the concrete graphical desktop interface can choose from among fieldsets, bullets, the same background colour or positioning the elements nearby vertically or horizontally. As a further example, it can choose to implement a navigator interactor through a graphical or a textual link or a button. When an interaction technique is selected then the associated attributes are enabled so that designers can specify the most suitable values. Thus, the concrete interface has a role of connecting high-level descriptions (the task model and the abstract interface) and implementations, which can be made in different implementation languages. This means that the concrete interface takes into account the features of the corresponding platform and there is a concrete interface for each possible platform.

Once the user interfaces (one for each potential user role) are generated, in order to make the cooperative interactive application work the developers have to associate the potential user interface events with the corresponding basic tasks in the task model. The basic tasks are those that cannot be further logically decomposed. Creating this association will allow the run-time support for the cooperative interface to have an updated view of the state of the application and use the task model to enable and disable the elements of the interfaces for the various users accordingly. To ease the creation of this association, Cooperative TERESA is able to automatically identify the events that can be generated in a user interface and may correspond to task performance, such as selection of images, buttons, or links, change of text fields and loading pages. Thus, it automatically lists, on one side, the basic tasks supported by the interface considered and, on the other side, the list of the potential events so that the designer only has to indicate their correspondences. Two different events may happen to correspond to one basic task (for example, when the selection of two different links has the same effect).

4. RUN-TIME SOFTWARE ARCHITECTURE

In our runtime software architecture for multi-device, multi-user Web applications we have one groupware server and various clients, which can be executed in different platforms (see Figure 2). At the implementation level the groupware server is obtained by adding some servlets and JSP pages in an Apache/Tomcat server, which also contains the application.

For each client, in addition to the application we have a JavaScript and an applet. The JavaScript has to:

- Capture the events that occur on the Web application and communicate them externally;
- Disable interface elements that correspond to basic tasks logically disabled according to the task model;
- Change the state of some interface elements in order to present information generated by other users.

The types of events that can be captured by the JavaScripts are clicks on images and elements of a form, changes in form elements, and the loading of Web pages. The communication between the JavaScripts and the external world is performed through an applet, which communicates with a servlet in the groupware server. The references to the applet and javascript are automatically added to the Web page upon its generation.

The groupware server is mainly composed of three components: the synchronizer manager, the interactive simulator and the event-task association table. The synchronizer manager is implemented as a Java Servlet, which can be called by the client applets and communicates with the content server. It can also provide the client applets with some information. The event-task associations table indicates which events should occur in order to perform each basic task. It plays an important role because it allows the environment to link the actual user behaviour with the semantic information contained in the task model. The interactive simulator is software that contains the task model of the cooperative application and is able to list the basic tasks enabled according to the temporal relations specified in the task model. When it receives notification that a basic task has been performed then its state is updated so that it can provide an updated list of enabled and disabled basic tasks.

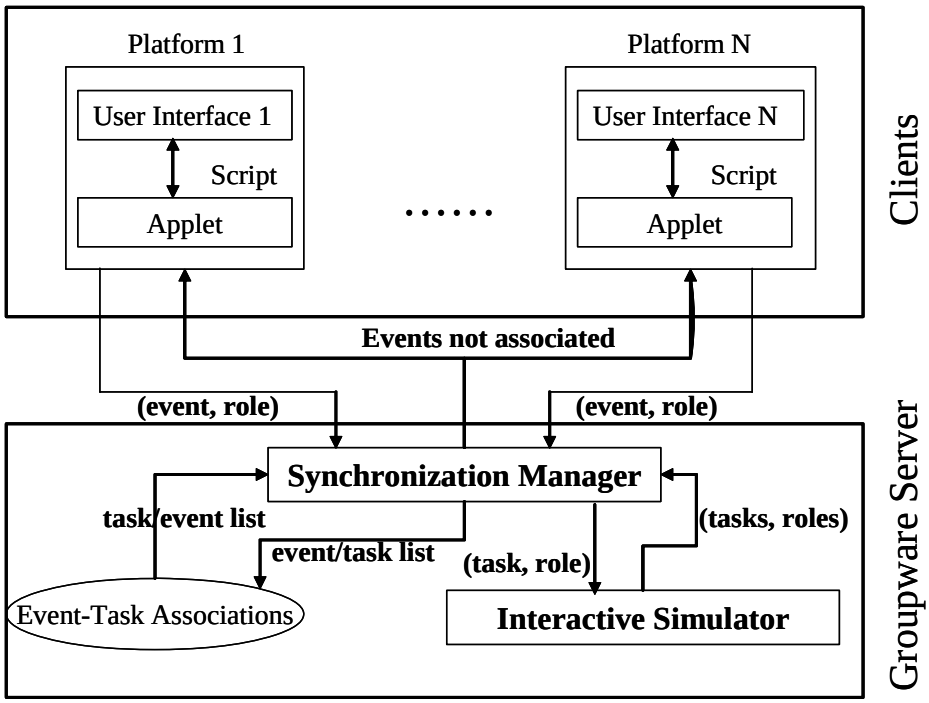

Figure 2. Run-Time Architecture for Cooperative TERESA.

At run-time, in order to coordinate the actions of users belonging to different roles while interacting with an application developed with Cooperative TERESA, it is necessary to capture the actions that each user performs. For this purpose each Web applications contains a script, which collects all the events performed by the users and sends them to the applet. The most important features sent to the servlet are the events performed by the user and the role that the user is playing. The groupware server of Cooperative TERESA is responsible for the coordination and cooperation between the roles. For this purpose it needs to know the basic tasks performed by the user. This is obtained by exploiting the association table created at design-time. After receiving information on the event performed, the Synchronizer Manager in the groupware server uses these associations to retrieve the corresponding task. If the corresponding task exists then it communicates it, along with the associated role, to the interactive simulator. If it does not exist, then the event does not correspond to any task and is ignored. The interactive simulator updates its state by performing the indicated basic task and identifying the updated list of enabled and disabled basic tasks for each role. This information is passed on to the synchronizer manager, which sends the corresponding list of disabled tasks to each client. In particular, this information is sent to the applet, which, in turn,

communicates it to the script, which disables the corresponding interface elements that are not enabled at that time.

The run-time support is also able to manage the exchange of information among various users so that if one user sends information to another one, such information will appear in the user interface of the other user according to the indications given at design-time. Indeed, there are interface elements that allow the user to send information, not to the content server, but to another user through the groupware server. In practise, Cooperative Teresa is able to associate specific attributes to buttons used to send information to other users. The JavaScript is able to recognise these attributes so that when the corresponding buttons are selected then the information to transmit externally is sent to the groupware server through the applet, with indications on the user who should receive it and to which interactor of the corresponding interface such information should be associated. Then, the server communicates with the applet of the target interface, which uses the local script to update the page content of the target interface, thus implementing our push mechanism.

5. EXAMPLE APPLICATION

Web-based learning applications offer flexibility and saving costs. Online courses can be taken in synchronous sessions. Taking into account this new learning trend, the example chosen to demonstrate Cooperative TERESA concerns an application that supports the cooperation and communication between professors and students. Thus, the roles involved are:

Student – After inserting personal data, the student has a number of options available, such as scheduling an appointment with the teacher, answering to questions on a number of topics, and entering the books and articles has read.

Professor – The professor has various options available: such as consulting appointments, after viewing all the details of the appointment s/he can send a confirmation or refuse the appointment, explaining the reason to the student; visualizing the student's answers, verifying them and submitting a comment to the student.

In the example implemented, the student role uses a desktop environment and the professor role a PDA because we assume that the teachers want to be able to accomplish their tasks from any location. For example, the student can choose a question from five different topics (see Figure 3, desktop interface). Upon the choice of the topic, a question and a set of four possible answers appear.

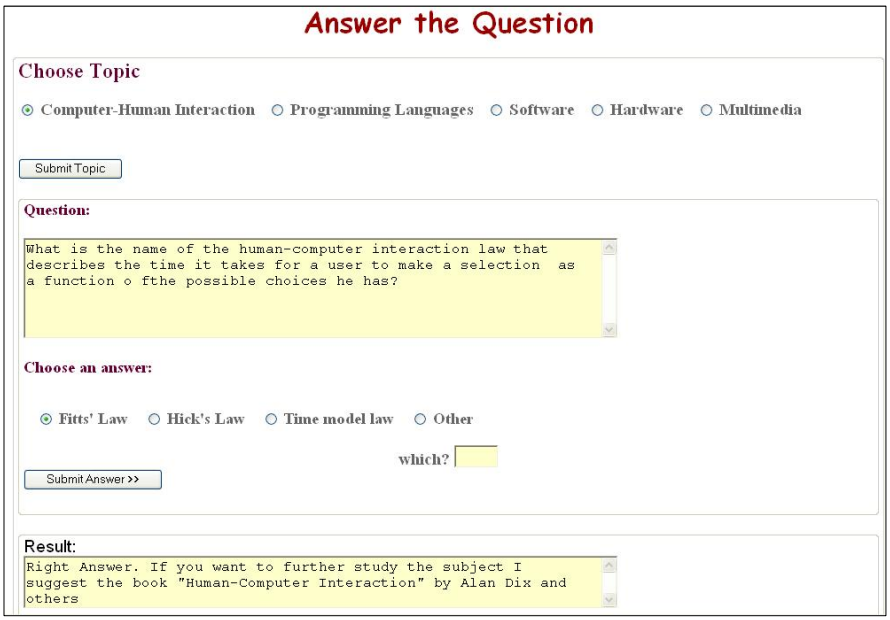

Figure 3. Example generated with Cooperative TERESA for desktop student interface.

After selecting one of the answers the student must press a button to send it. This action disables all the previous actions, changing the fields to a grey colour, and sends the student's selected question and answer to the professor. Upon the submission of the answer by the student, the professor is now able to access the "Student's Performance" link. In the Web page that appears, in the top part the professor can analyze information about the student (sent by the student upon the log in), the question and the answer, and in the bottom part indicate to the student if the answer is correct or incorrect, including a comment (see Figure 3 PDA interface). Then, the student can immediately visualize the professor's comment.

Another possibility of interaction between professor and student is the scheduling of an appointment with the professor. The student can schedule an appointment by sending the date, the hour and the motive of the appointment. After visualizing the appointment information, the professor can choose if he/she is available. If not available, the professor writes the reason and submits it to the student. If available, the professor simply sends the date and time back to the student as confirmation. After, the student can see if the professor will be at the appointment, and if not, why. The example chosen covers the communication and cooperation aspects of multi-user applications.

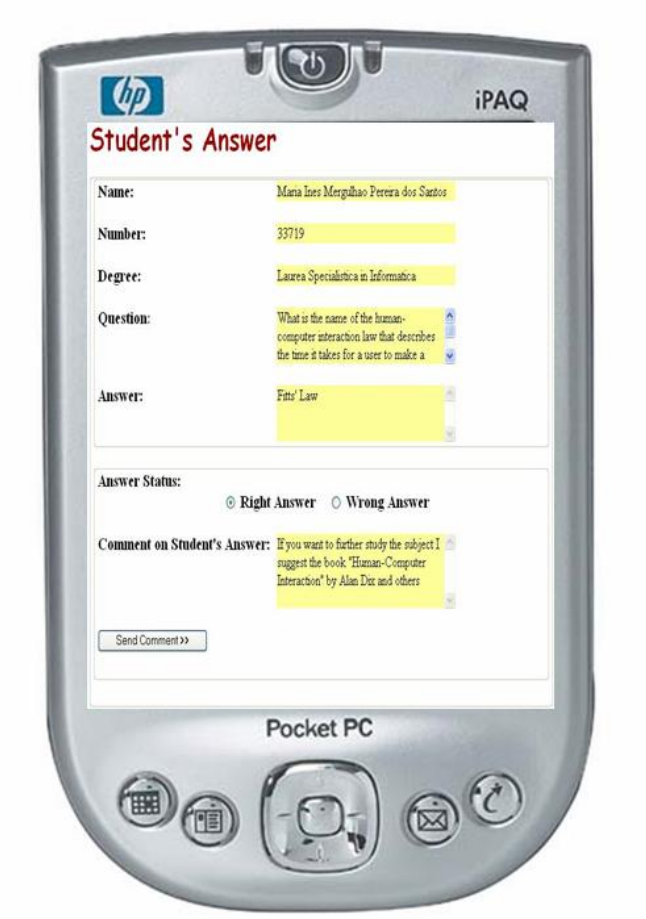

Figure 4. Example generated with Cooperative TERESA for PDA teacher interface.

6. CONCLUSIONS AND FUTURE WORK

We have presented an environment able to support design, development and run-time support for multi-user, multi-device Web applications. Our prototype has been validated with the development of an application for e-learning supporting cooperation between students and teachers. This has shown the feasibility of the approach and its possible advantages: the environment actually allows designers to focus on the tasks to support, the relations among tasks performed by different users, and the logical structure of the corresponding user interface without having to manage a plethora of low-level implementation details. Then, a specific run-time architecture is able to support even synchronous communication among users interacting

through different types of devices. Thus, the Web interface of one user can be dynamically modified to present information generated by other users.

Future work will be dedicated to extending the set of possible platforms for the user interfaces involved in the multi-user application, also considering various modalities, such as vocal, gestural and graphical modality, and different ways to combine them. This means integrating into Cooperative Teresa the already existing transformations: from abstract TERESA XML first to the concrete descriptions of the corresponding platforms and then to the final interfaces in implementation languages able to support such modalities.

7. REFERENCES

[1] Girgensohn A. and Lee, A., Developing Collaborative Applications On the World Wide Web, CHI 98 conference summary on Human factors in computing systems, 141-142, ACM Press, April 1998

[2] Trætteberg, H., Modeling work: Workflow and Task modeling. In: Vanderdonckt, J., Puerta, A.R. (eds.): Proc. of 3 rd Conf. on Computer-Aided Design of User Interfaces CADUI'99 (Louvain-la-Neuve, 21-23 October 1999). Kluwer Academics, Dordrecht (1999) 275–280.

[3] Pinelle, D., Gutwin, C., Greenberg, S. Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration Pages: 281 – 311, ACM Transactions on Computer-Human Interaction, December 2003.

[4] Paternò, F. Model-based Design and Evaluation of Interactive Applications. Springer Verlag, ISBN 1-85233-155-0, 1999.

[5] Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., Shuster, J., 1994. UIML: An Appliance-Independent XML User Interface Language, Proceedings of the 8th WWW conference.

[6] Mori, G. , Paternò, F., and Santoro, C., Design and Development of Multi-Device User Interfaces through Multiple Logical Descriptions, IEEE Transactions on Software Engineering, August 2004, Vol.30, N.8, pp.507-520, IEEE Press.

[7] Stanciulescu A., Limbourg Q., Vanderdonckt J., Michotte B., Montero F., A Transformational Approach for Multimodal Web User Interfaces based on USIXML. Proceedings ICMI 2005, pages 259-266, ACM Press.

[8] Roseman, M. and Greenberg, S. (1996). Building Real Time Groupware with GroupKit, A Groupware Toolkit. March. ACM Transactions on Computer Human Interaction, 3(1), p.66-106, ACM Press.

[9] Han, R., Perret, V., Naghshineh, M., "WebSplitter: Orchestrating Multiple Devices for Collaborative Web Browsing", ACM Conference on Computer Supported Cooperative Work (CSCW), December 2, 2000, Pages: 221 – 230.

[10] Greenberg, S. Toolkits and Interface Creativity, Special Issue on Groupware, Journal Multimedia Tools and Applications, Kluwer.

[11] Grundy, J. , Wang X. and Hosking, J. , Building Multi-device, Component-based, Thin-client Groupware: Issues and Experiences, Australian Computer Science Communications, Third Australasian Conference on User Interfaces, Volume 7 CRPITS '02, 71-80, Australian Computer Society, Inc., January 2002.