Authoring Interfaces with Combined Use of Graphics and Voice for both Stationary and Mobile Devices

Fabio Paternò, Federico Giammarino
ISTI-CNR
Via G.Moruzzi 1
56124 Pisa, Italy
+390503153066
{fabio.paternò, federico.giammarino}@isti.cnr.it

ABSTRACT
The technological evolution is making multimodal technology available to the mass market with increased reliability. However, developing multimodal interfaces is still difficult and there is a lack of authoring tools for this purpose, especially when multi-device environments are addressed. In this paper, we present a method and a supporting tool for authoring user interfaces with various ways to combine graphics and voice in multi-device environments. The tool is based on the use of logical descriptions and provides designers and developers with support to manage the underlying complexity, make and modify design choices, and exploit the possibilities offered by multimodality.

Categories and Subject Descriptors
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Design, Human Factors, Languages.

Keywords
Multimodal Interfaces, Authoring Environments, Web, Graphical and vocal modalities, X+V.

1. INTRODUCTION
The increasing availability of various types of interactive devices, supporting various combination of modalities, raises a number of issues for designers and developers of interactive applications. The problem is how to help them to develop various versions of their interfaces that are able to adapt to the various interaction resources available and avoid confusing the designer by the many details related to the various devices and implementation languages.

To address such issues there has been renewed interest in model-based design of user interfaces. The idea is to have some logical descriptions close to the user's view and then intelligent environments able to transform them in order to obtain interfaces adapted to the target devices. A number of XML-based proposals in this area have been put forward (such as TERESA [6], XIML[10], UsiXML [11], UIML [1], Pebbles [7]). Such approaches have focused on mono-modal user interfaces (either graphical or vocal). Thus, the main adaptation was usually to change the presentation, content, and navigation to suit the various possible sizes of the graphical screen or the vocal structure.

However, this is not enough. Recent technological evolutions are making available to the mass market various interaction modalities (such as vocal and gestural interaction) and allow various combined use of such modalities with the graphical one. Developing multimodal user interfaces is still difficult. Indeed, even if we consider the Web, most authoring tools (such as MacroMedia Dreamware) only support graphical interfaces. Developing multimodal interfaces in multi-device environments is even more difficult, because identification of the most suitable ways to combine the modalities has to take into account the features of the hosting device and the potential contexts in which it will likely be used.

In the paper we first introduce related work and the starting point of this work, the previous environment for monomodal interfaces [6]. We move on to describe how the new version for multimodal interfaces has been designed and show the resulting authoring environment. We then describe an example application. Lastly, we report on first experiences by designers, draw some conclusions and discuss future work that can exploit the generality of the approach in order to support interfaces involving other modalities (such as gestural and tactile interaction).

2. RELATED WORK
Obrenovic et al. [12] have investigated the use of conceptual models expressed in UML in order to then derive graphical, form-based interfaces for desktop or mobile devices or vocal devices. UML is a software engineering standard mainly developed for designing the internal software of application

functionalities. Thus, it seems unsuitable to capture the specific characteristics of user interfaces and their software. The ICO formalism for user interfaces has shown to be suitable to model and specify multimodal interfaces mainly for analysis in safety-critical application [2], and it has limited support for generation of multi-modal interfaces from such specifications

One interesting effort to ease multimodal interface development is ICARE [3]: it provides a graphical environment for a component-based user interface exploiting various modalities and modules that allow various compositions of such modalities. In this paper we present a different approach: we show how we can derive multimodal interfaces starting with logical descriptions of tasks and user interfaces obtained through general, platform-independent notations. We still provide the possibility of combining the modalities in various ways, but at different granularity levels (inside a single interaction object and among several interaction objects). While some other work has been carried out to apply transformations to logical descriptions to derive multimodal interfaces [11], our work has been able to provide an authoring environment that is able to suggest solutions for identifying how to combine various modalities and allows designers to easily modify them in order to tailor the interface generation to specific needs. This result has been obtained by extending a previously existing authoring tool [6] that was limited to creating only graphical or vocal interfaces.

3. BACKGROUND

In the research community in model-based design of user interfaces there is a consensus on what the useful logical descriptions are [4][9][12]:

- The task and object level, which reflects the user view of the interactive system in terms of logical activities and objects that should be manipulated to accomplish them;
- The abstract user interface, which provides a modality independent description of the user interface;
- The concrete interface, which provides a modality dependent but implementation language independent description of the user interface;
- The final implementation, in an implementation language for user interfaces.

Thus, for example we can consider the task *select a work of art*, this implies the need for a selection object at the abstract level which indicates nothing regarding the modality in which the selection will be performed (it could be through a gesture or a vocal command or a graphical interaction). When we move to the concrete description then we have to assume a specific modality, for example the graphical modality, and indicate a specific modality-dependent interaction technique to support the interaction in question (for example, selection could be through a radio-button or a list or a drop-down menu), but nothing is indicated in terms of a specific implementation language. When we choose an implementation language we are ready to make the last transformation from the concrete description into the syntax of a specific user interface implementation language.

The advantage of this type of approach is that it allows designers to focus on logical aspects and take into account the user view right from the earliest stages of the design process. In the case of interfaces that can be accessed through different types of devices the approach has additional advantages. First of all, the task and the abstract level can be described through the same language for whatever platform we aim to address. Then, in our approach we have a concrete interface language for each target platform. By platform we mean a set of interaction resources that share similar capabilities (for example the graphical desktop, the vocal one, the cellphone, the graphical and vocal desktop). Thus, a given platform identifies the type of interaction environment available for the user, and this clearly depends on the modalities supported by the platform itself. Actually, in our approach the concrete level is a refinement of the abstract interface depending on the associated platform. This means that all the concrete interface languages share the same structure and add concrete platform-dependent details on the possible attributes for implementing the logical interaction objects and the ways to compose them. All languages in our approach, for any abstraction level, are defined in terms of XML in order to make them more easily manageable and allow their export/import in different tools.

Another advantage of this approach is that maintaining links among the elements in the various abstraction levels allows the possibility of linking semantic information (such as the activity that users intend to do) and implementation levels, which can be exploited in many ways. A further advantage is that designers of multi-device interfaces do not have to learn the many details of the many possible implementation languages because the environment allows them to have full control over the design through the logical descriptions and leave the implementation to an automatic transformation from the concrete level to the target implementation language. In addition, if a new implementation language needs to be addressed, the entire structure of the environment does not change, but only the transformation from the associated concrete level to the new language has to be added. This is not difficult because the concrete level is already a detailed description of how the interface should be structured.

4. MULTIMODAL TERESA

The goal of Multimodal TERESA is to provide an authoring environment for flexible development of multimodal interfaces in multi-device environments. Our first multimodal target environment provides for composition of graphical and vocal interactions. There are many ways to compose such modalities. The idea is to provide a structured support that aims to identify the most suitable solutions at various granularity levels. By default the tool provides some specific solutions that can be modified by the designers to suit their specific needs. In terms of target implementation languages, the first supported is X+V [13] because it supports multimodality through the Web, which is the most common interaction environment, it is a standard and currently some publicly available browsers (such as Opera) support it, thus allowing developers to immediately test the resulting interfaces.

In this paper we focus on how the tool provides such support for multimodal interfaces. As we introduced in the previous section, the task and the abstract level are described by modality-independent languages. Then, we have a concrete language for

supporting each target platform. Thus, we have a concrete language for each platform. It is important to remember that the concrete languages are just refinements of the abstract languages. In the case of the graphical+vocal desktop platform we have structured our authoring environment in three sections: one dedicated to specifying the concrete attributes, one for the vocal attributes and one for indicating in a logical manner how to compose them.

As schematically described in the example in Figure 1, the abstract user interface is structured into a number of presentations, each of which contains instances of logical interaction objects classified depending on their semantics (selection, multi-selection, edit, navigator, description, …) and instances of composition operators that indicate how to put together the various interface elements such as grouping (highlighting that a set of objects are logically related to each other) or hierarchy (indicating that there are various levels of importance for the interface elements involved). The purpose of the navigator elements is to indicate when and how to move from one presentation to another.

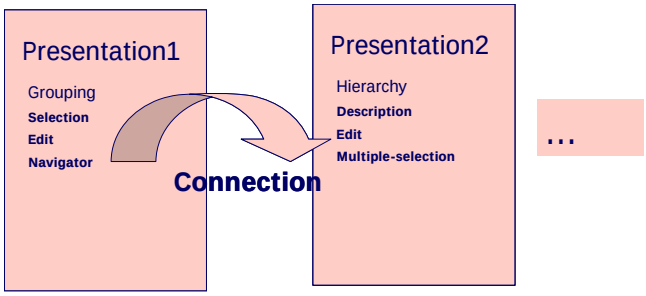

Figure 1. Example of Abstract Interface Structure.

In order to indicate how to combine the modalities we have considered four well-known properties (CARE: Complementarity, Assignment, Redundancy, Equivalence) [5] at various granularity levels. We apply such properties in the following manner:

- complementarity, the considered part of the interface is partly supported by one modality and partly by another one;
- assignment, the considered part of the interface is supported by one assigned modality;
- redundancy, the considered part of the interface is supported by both modalities;
- equivalence, the considered part of the interface is supported by either one modality or another.

Since we want to provide a flexible environment, we support the possibility of applying such properties in the implementation of the various aspects characterising our logical descriptions: the composition operators, the interaction and the only-output elements. In addition, in order to have the possibility of controlling multimodality at a finer level, the interaction elements are structured into three stages:

- Prompt: represents the interface output indicating that it is ready to receive an input.
- Input: represents how the user can actually provide the input.
- Feedback: represents the response of the system after the user input.

Table 1. How CARE Properties have been made available for graphical+vocal desktop and graphical+vocal mobile..

Element type	*Interaction phase*	*CARE Property for Desktop*	*CARE Property for PDA*
Composition Operators			
Grouping Hierarchy Ordering	Output	Graphical Assignment Redundancy	Graphical Assignment Redundancy
	Feedback	Vocal Assignment	Vocal Assignment
Only output Interactors			
Text Description	Output	Graphical Assignment Redundancy Complementarity	Vocal Assignment Redundancy Complementa.
Interaction Interactors			
Numerical/ Text Edit Single/ Multiple Selection	Input	Graphical Assignment Equivalence	Graphical Assignment Equivalence
	Prompt	Graphical Assignment Redundancy	Graphical Assignment Redundancy
	Feedback	Graphical Assignment Redundancy	Graphical Assignment Redundancy
Navigator Activator	Input	Graphical Assignment Equivalence	Graphical Assignment Equivalence
	Prompt	Graphical Assignment Redundancy	Graphical Assignment Redundancy

Thus, our environment allows the designer to decide what multimodal support to provide for each of the different stages.

How such properties will be applied to such elements depends on the modalities and platforms considered. In the following we describe how they are applied to the vocal+graphical interfaces, but the approach is general and can be applied to any types of modalities.

In order to avoid confusing the designer, the environment provides a default suggestion for each element at each level, with the possibility to change it with other meaningful choices. The possible choices have been identified taking into account the features of the target platforms. Thus, in the case of the multimodal desktop, in which the graphical resources are rich, then we have the composition operators supported graphically. The interaction elements are structured in such a way that the prompt is graphical, input can be either graphical or vocal, and feedback is in both modalities. The only-output elements are graphical. In the multimodal PDA, in which the graphical resources are less rich, we have the composition operators supported both graphically and vocally, the interaction elements are supported in such a way that the prompt is both vocal and graphical, the input either graphical or vocal, and feedback in both modalities. The only-output elements are either both graphical and vocal or they use the two modalities in a complementary way if they take a lot of resources.

Table 1 provides details on how the CARE properties are applied in the generation of graphical and vocal interfaces for desktop and PDA platforms. It also shows what properties have been deemed meaningful in the case of graphical and vocal interfaces, and thus are supported in the authoring environment. While they are similar for the two types of platforms, there are differences in the default properties applied by the environment, taking into account the richer set of graphical resources of the desktop platform and that the mobile device can be often used on the move. In Table 1, the first column indicates the element of the abstract interface considered. Depending on the element various interaction phases (input, prompt, feedback) have to be considered. The composition operators aim to put together some interface elements in such a way to highlight logical closeness or hierarchy of importance or some ordering. Thus, usually there is some output information to indicate the involved elements (for example, it could be a graphical container or a sound at the beginning and the end of the grouped elements) and there may be some feedback when one of the composed elements changes its state. The navigator allows the user to move from one point to another of the application. This type of element usually has no immediate feedback because the actual feedback is given by the change of the application presentation loaded.

5. THE AUTHORING ENVIRONMENT

The resulting authoring environment allows designers and developers to start from two possible points: the task model description or the abstract interface description. In both cases they have to specify the target platform (in the current tool version either multimodal desktop or multimodal PDA). If they start with the task model then the tool automatically generates the corresponding abstract interface (see Figure 2 for an example). As you can see the main area is mainly divided into four parts: the top-left dedicated to the list of presentations composing the user interface, the bottom-left indicating the connections defining how it is possible to move from one presentation to another, the top-right indicating the abstract description of the currently selected presentation and the bottom-right part displays the description of the possible concrete implementation of the currently selected element in the abstract part. In the next section we provide more detail using an example application.

The concrete part has three tabbed panes, one for the concrete graphical attributes, one for the concrete vocal attributes and one to specify how to compose the multimodal attributes. Table 1 shows the list of the possible ways to compose the modalities that have been deemed meaningful.

For example, if we consider the single selection interactor used to indicate the time of a cinema reservation, then the tool as a first suggestion for an implementation in a graphical+vocal desktop interface would propose that the input be equivalent (either graphical or vocal) and the prompt and feedback both redundant. Then, in the vocal section there would be an indication of the corresponding label, and the associated vocal message and feedback, in addition to the definition of the possible choice elements. The graphical part indicates the interaction technique for implementation (e.g. a radio-button), and the corresponding label and elements. The tool keeps information in the graphical and vocal parts consistent. So, if the designer indicates five possible choice elements in the vocal part, then this is indicated when the graphical part is accessed as well. Likewise, in the case of a text output, if the corresponding multimodal property is complementarity, then different texts can be specified for vocal and graphical rendering, while if the multimodal attribute is redundant, then the text modified in either part will be updated for the other one as well.

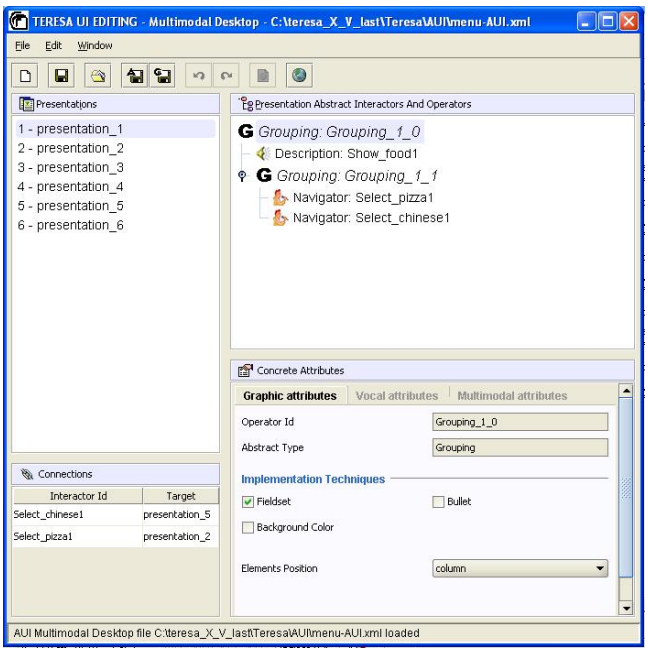

Figure 2. Our Authoring Environment.

6. EXAMPLE APPLICATION

A number of applications have been developed to test the tool. One concerns a cinema application. When users access the home page they can first select a town, then they can choose the movie that they want to see, lastly they can make a reservation indicating the preferred seat.

Figure 3 shows the tool while authoring such an application, in particular the logical description of the interface. This can be obtained either by editing it through direct manipulation techniques or as a result of an automatic transformation from the task model description.

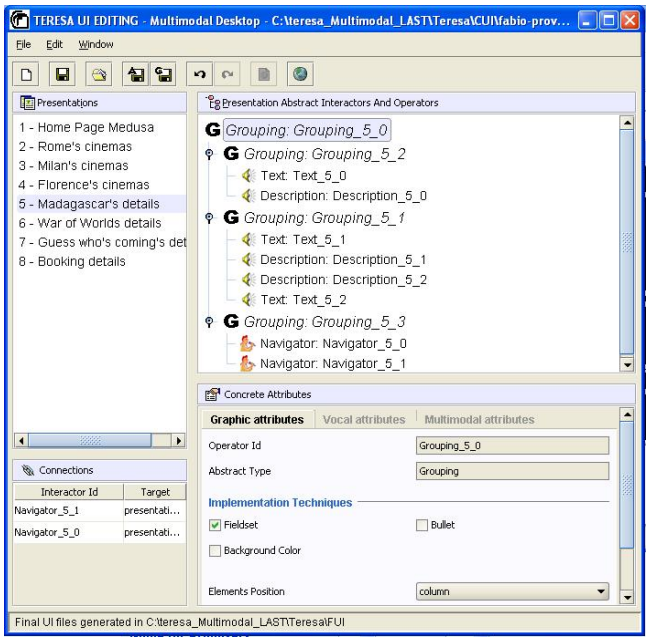

Figure 3. The Authoring Tool with the Example Application.

In the case of automatic transformation from the task model, then the designer still has to edit the values of the concrete attributes in order to make the resulting user interface more appealing. The top-left of Figure 3 shows the list of the defined presentations. The currently selected one concerns the movie "Madagascar". We can see that the abstract part is structured through some grouping operators. One groups one text and one description element, another one groups two text and two description elements. The last grouping refers to the two navigational elements, indicating that they should be lined up horizontally. The main grouping element puts together all the groupings indicating that they should be organized vertically (column attribute value) and using a fieldset (usually implemented through a rectangle) to group all the involved elements. Figure 4 shows the interface automatically generated from the logical description in Figure 3.

The text and the description elements are obtained through the graphical modality. The description consists in text with the support of some images. The two navigator elements allow the user to move to the reservation part or to the home page. The input for selecting the next page to access can be provided equivalently through either the vocal or graphical modality. The prompt is given by the label of the two corresponding buttons. In addition, for highlighting such possibilities, a redundant vocal prompt is given ("Say *book* to buy seats for the movie or *back* to return to the home page").

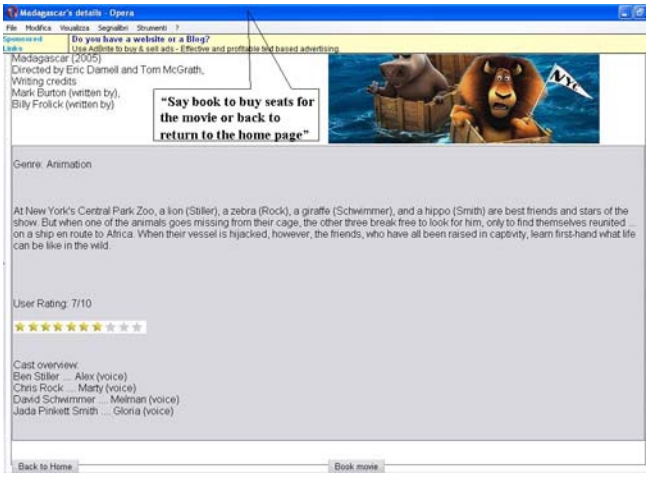

Figure 4. The Desktop Multimodal Interface Generated.

Figure 5 shows how this type of presentation is generated for a mobile device. In this case, while the logical structure of the page is still the same, there are changes on how the multi-modality is supported. The vocal modality is much better exploited because of the limited graphical resources. Thus, some information is provided only vocally, some is provided both vocally and graphically and some is provided by exploiting the two modalities in a complementary way.

Figure 5. The corresponding Mobile Interface.

Figure 6 shows the logical structure of the interface supporting the reservation form. There is an internal grouping for the editing of the various fields (one text edit and three single selections), a further grouping for the two navigational elements and an external grouping that combines it with some introductory text and two navigator elements for moving to other parts of the application. The currently selected element is a single selection object and in the multimodal attributes panel, in the right-bottom part, we can see the currently selected choices and the possible alternatives that the designer can easily select. Currently, input is set to equivalence (the user can make the selection either graphically or vocally) and prompt and feedback are set to redundancy (they are rendered both graphically and vocally) but the designer can choose a only graphical implementation through the associated radio-button and then the generated pages will be immediately updated to the new choice.

Figure 7 shows the resulting interface. The introductory part is provided complementarily by vocal and graphical modality. All the form fields have redundant prompts and feedback and can be entered in either the vocal or graphical modality. In order not to overload the vocal channel the navigator elements have only graphical prompt, but they can still be selected using either modality.

7. THE MULTIMODAL INTERFACE IMPLEMENTATION GENERATION

Currently the tool generates multimodal implementation in X+V. This is a W3C standard already supported by freely available browsers such as Opera. Thus, it is possible for all users to access the graphical and vocal interfaces generated. In addition, the support for other implementation languages can be introduced with limited effort. Indeed, this would require simply modifying the transformation from the concrete interface description to the target implementation language. This is mainly obtained through mappings between constructs in the two languages.

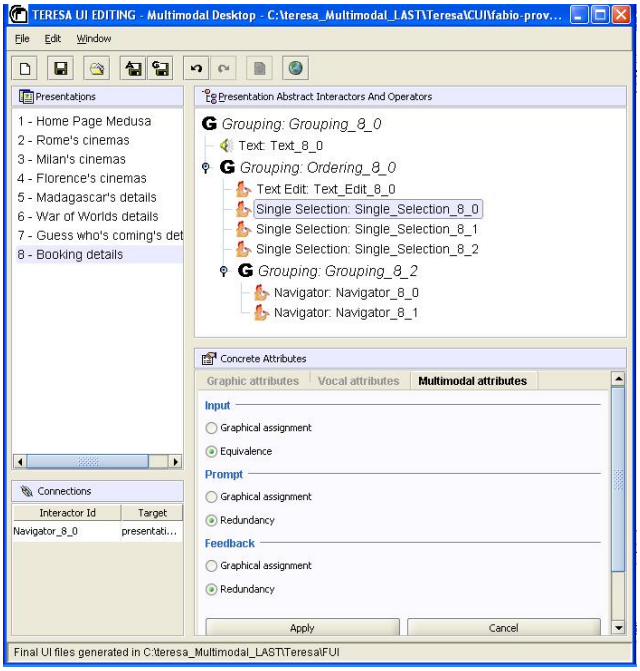

Figure 6: The Authoring of the Multimodal Form.

The generated X+V files are divided into three parts. The heading indicates the XML version, the DOCTYPE and the DTD of the language. Then, the tag <html> is open to indicate the modules used and contains the head and body. The second part is the head, which includes all the vocal functions and defines the page title and indicates the CSS files to use. The vocal functions are contained in the tag <vxml:form> , which contains all the vocal constructs corresponding to the elements composing the concrete interface. The third part is the body, which contains all the graphical HTML constructs corresponding to the elements in the concrete language. In addition, it contains a reference, in the form of event handler, to the tag <vxml:form> which manages the vocal part.

In the generation of the vocal part, our authoring environment is also able to take into account the designers' specifications and generate the grammars indicating the various possible combinations of vocal input that the application can accept for the vocal interactions.

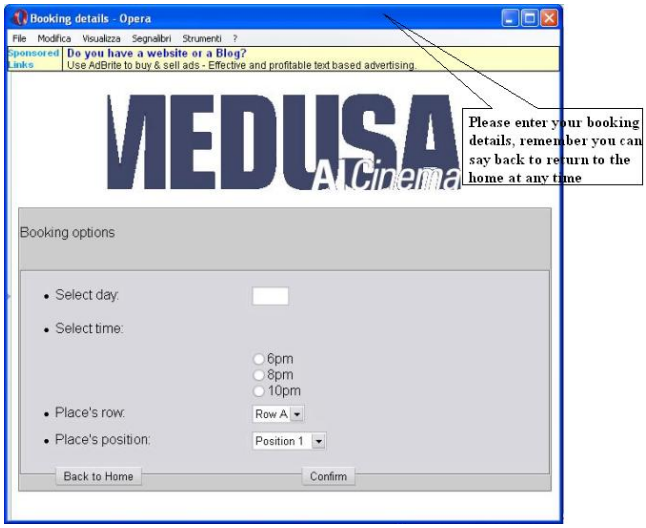

Figure 7. The Resulting Multimodal Form.

8. PRELIMINARY EVALUATION

We performed a test to check whether the tool can be actually used by people without experience in multimodal interface programming. In a EU Research Training Network it was organised a seminar which was attended by 5 senior and 7 young researchers with different backgrounds: formal modelling, cognitive psychology, informatics, engineering. None of them had previous experience in multimodal interface programming. After a short introduction on X+V (30 minutes) and on the authoring environment (30 minutes) they were asked to develop a multimodal interface using the tool. It was interesting to see that after one hour they were actually able to obtain some multimodal interfaces. The tool at that time was still under development and so some small bugs were identified during the exercise but the interesting point was that all participants were able to design some multimodal pages using the tool. During the exercise one of the authors was available to provide help but this was mainly limited to identify the most suitable navigational structure of the resulting application for supporting the desired tasks.

Further empirical tests with the authoring environment will be carried out in the future. We will make the tool publicly available soon for stimulating further feedback and comments on its features.

9. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an authoring environment for multimodal interfaces. It allows designers to work through logical descriptions of the user interface and provides support for choosing the most suitable combination of various modalities at different granularity levels and for the various parts of the user interface.

Future work will be dedicated to extending the environment in order to provide support for additional modalities, such as tactile and gestural interaction, in several possible combinations, still for both stationary and mobile devices. Thus, it is possible to obtain an universal authoring environment, based on the use of logical device independent languages able to generate interfaces that adapt to varying interaction modalities.

10. REFERENCES

[1] 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.

[2] Bastide, R., Navarre, D., Palanque, P., Schyn A. & Dragicevic, P., A Model-Based Approach for Real-Time Embedded Multimodal Systems in Military Aircrafts, Proceedings ICMI 2004, pages 243-250, ACM Press.

[3] Bouchet, J., Nigay, L. Ganille T., ICARE software components for rapidly developing multimodal interfaces. Proceedings ICMI 2004, pages 251-258, ACM Press.

[4] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers. Vol. 15, No. 3, June 2003, pp. 289-308.

[5] Coutaz J., Nigay L., Salber D.,.Blandford A, May J., Young R., 1995. Four Easy Pieces for Assessing the Usability of Multmodal Interaction: the CARE properties. Proceedings INTERACT 1995, pp.115-120.

[6] Mori, G. Paternò, F. 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] Nichols, J. Myers B. A., Higgins M., Hughes J., Harris T. K., Rosenfeld R., Pignol M., 2002. "Generating remote control interfaces for complex appliances". Proceedings ACM UIST'02, pp.161-170.

[8] Obrenovic, Z., Starcevic D., Selic B., A Model-Driven Approach to Content Repurposing, IEEE Mutimedia, Januray March 2004, pp.62-71.

[9] Paternò, F., "Model-Based *Design and Evaluation of Interactive Application"*. Springer Verlag, ISBN 1-85233-155-0, 1999.

[10] Puerta, A., Eisenstein, XIML: A Common Representation for Interaction Data, Proceedings ACM IUI'01, pp.214-215.

[11] 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.

[12] Szekely, P., 1996. Retrospective and Challenges for Model-Based Interface Development. 2nd International Workshop on Computer-Aided Design of User Interfaces, Namur, Namur University Press.

[13] W3C XHTML+Voice Profile 1.0, http://www.w3.org/TR/xhtml+voice/