Supporting Multimodality in Service-oriented Model-Based Development Environments

Marco Manca, Fabio Paternò

Supporting Multimodality in Service-oriented Model-based Development Environments

Marco Manca, Fabio Paternò

CNR-ISTI, HIIS Laboratory, Via Moruzzi 1, 5614 Pisa, Italy
{Marco.Manca, Fabio.Paterno}@isti.cnr.it

Abstract. While multimodal interfaces are becoming more and more used and supported, their development is still difficult and there is a lack of authoring tools for this purpose. The goal of this work is to discuss how multimodality can be specified in model-based languages and apply such solution to the composition of graphical and vocal interactions. In particular, we show how to provide structured support that aims to identify the most suitable solutions for modelling multimodality at various detail levels. This is obtained using, amongst other techniques, the well-known CARE properties in the context of a model-based language able to support service-based applications and modern Web 2.0 interactions. The method is supported by an authoring environment, which provides some specific solutions that can be modified by the designers to better suit their specific needs, and is able to generate implementations of multimodal interfaces in Web environments. An example of modelling a multimodal application and the corresponding, automatically generated, user interfaces is reported as well.

Keywords: Multimodal interfaces. Model-based design, Authoring tools.

1 Introduction

Multimodal user interfaces support various user input modes. Ongoing technological evolution is making such interfaces more and more affordable and is proposing them in the mass market as well. However, developing multimodal user interfaces is still difficult and there is a lack of authoring environments for this purpose.

Model-based approaches have received renewed attention in recent years because they can help developers in managing the complexity of designing and developing multi-device applications. Most of the proposed model-based approaches have focused on desktop and mobile applications, sometimes with support for vocal interfaces as well, but there has been little effort in applying them to multimodal user interfaces, and such rare studies have found limited applications, as results were still too preliminary to provide general solutions.

In this paper, we present a logical language and an associated authoring environment able to provide a useful and general solution to such issues, and which can be exploited by developers of multimodal user interfaces. In the paper after discussing related work we introduce our approach to modelling multimodal interaction; next we show how it has been formalized in an XML logical language to address composition of vocal and graphical modalities, and we present how such language is supported within an authoring environment. Then, the transformation from the logical description to implementation is discussed, and an example multimodal application obtained through this environment is presented as well. Lastly, some conclusions are drawn along with indications for future work.

2 Related Work

The problem of designing multi-modal interfaces has been addresses in some previous work but still needs more general and better engineered solutions. Damask [7] includes the concept of layers to support the development of cross-device (desktop, smartphone, voice) user interfaces. Thus, the designers can specify user interface elements that should belong to all the user interface versions and elements that should be used only with one device type. However, this approach can be useful in developing single modality versions (graphical or vocal) but does not provide particularly useful support when considering multimodal interfaces, which require specific support to indicate how to compose the involved modalities. XFormsMM [5] is an attempt to extend XForms in order to derive both graphical and vocal interfaces. In this case the basic idea is to specify the abstract controls with XForms elements and then use aural and visual CSS for vocal and graphical rendering, respectively. The problem in this case is that aural CSS have limited possibilities in terms of vocal interaction and the solution proposed requires a specific ad hoc environment in order to work. For this purpose we propose a more general solution able to derive different implementations for desktop and mobile devices. Obrenovic et al. [11] 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 ones. 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. In [15] there is a proposal to derive multimodal user interfaces using attribute graph grammars, which have a well-defined semantics but limitations in terms of performance. The possibility of deriving mutlimodal interfaces was addressed in [12] but using hardcoded solutions for the transformation and logical descriptions that were unable to describe typical Web2.0 interactions and access to Web services.

A different approach to multimodal user interface development has been proposed in [6], which aims to provide a workbench for prototyping them using off-the-shelf heterogeneous components. In that case model-based descriptions are not used and it is necessary to have an available set of previously defined components able to communicate through low-level interfaces, thus making it possible for a graphical editor to easily compose them.

To summarise, we can say that the few research proposals that have also considered multimodal interfaces have not been able to obtain a general solution in terms of logical descriptions and provide limited support in terms of generation of the corresponding user interface implementations. For example, in [12] the transformations were hard-coded in the Java implementation, while in [15] the transformations were specified using attributed graph grammars, whose semantics is formally defined but have considerable performance limitations.

In this paper we present a general logical language for multimodal interaction, which is included in an overall environment able to support development of multi-device user interfaces. The associated authoring environment includes a transformation tool able to derive X+V implementations from the logical specifications and satisfies the requirements for multimodal interface generation discussed in previous work [10], such as modality independence, support for specifying hierarchical grouping, etc.

3 Background

MARIA [13] is a recent model-based language, which allows designers to specify abstract and concrete user interface languages according to the CAMELEON Reference framework [2]. This language represents a step forward in this area because it provides abstractions also for describing modern Web 2.0 dynamic user interfaces and Web service accesses. In its first version it provides an abstract language independent of the interaction modalities and concrete languages for graphical desktop and mobile platforms. In general, concrete languages are dependent on the typical interaction resources of the target platform but independent of the implementation languages. In this paper we present a concrete language for multimodal interfaces, which has been designed within the MARIA framework.

In MARIA an abstract user interface is composed of one or multiple presentations, a data model, and a set of external functions. Each presentation contains: a number of user interface elements (interactors) and interactor compositions (indicating how to group or relate a set of interactors); a dialogue model, describing the dynamic behaviour of such elements and connections, indicating when a change of presentation should occur. The interactors are classified in abstract terms, e.g. edit, selection, output, control. Each interactor can be associated with a number of event handlers, which can change properties of other interactors or activate external functions. While in graphical interfaces the concept of presentation can be easily mapped on that of a set of user interface elements perceivable at a given time (e.g. a page in the Web context), in the case of a vocal interface we consider a presentation as a set of communications between the vocal device and the user that can be considered as a logical unit, e.g. a dialogue supporting the collection of information regarding a user. In defining the vocal concrete language [14] we have refined the abstract vocabulary for this platform. This mainly means that we have defined vocal refinements for the elements defined in the abstract language: interactors (user interface elements), the associated events and their compositions. The multimodal support has been built on top of such parts following an approach discussed in the next section.

4 Approach to Modelling Multimodal Interaction

In this paper we present a multimodal environment able to support composition of graphical and vocal interactions. There are many ways to compose such modalities. The goal is to provide a structured support that aims to identify the most suitable solutions at various granularity levels. In order to indicate how to combine the modalities, we have considered the well-known CARE properties (CARE: Complementarity, Assignment, Redundancy, Equivalence) [4] at various granularity levels. We apply such properties in the following manner:

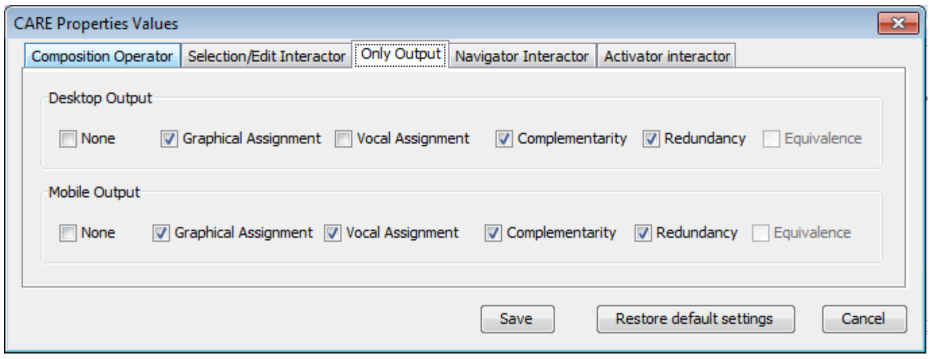
- *Complementarity*: the considered part of the interface is partly supported by one modality and partly by another one;
- *Assignment*: the considered part of the interface is supported by one assigned modality;
- *Redundancy*: the considered part of the interface is supported by both modalities;
- *Equivalence*: the considered part of the interface is supported by either one modality or another.

How such properties will be applied to the user interface elements depends on the modalities and platforms considered. In the following, how these properties are applied to mixed vocal+graphical interfaces in both desktop and mobile devices is described, but the approach presented can be applied to other types of modalities.
Since we want to provide a flexible environment, the possibility of applying such properties is supported in the definition 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 phases (each of them can be associated with a different CARE property):

- 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.

In practise, not all the CARE properties can be applied to all the three phases of an interaction. In particular, equivalence can be applied only to input: when two modalities are available and either one or the other can be used to enter the input. Vice versa, redundancy can be applied to prompt and feedback, but not to input, since a redundant input would mean that the same input is provided through different modalities, which does not seem useful or efficient. Complementarity could be applied to all three phases. However, in the case of input it can meaningfully be applied when structured input are considered. Indeed, atomic inputs that require simple actions (e.g. button selection) can hardly be obtained through a complementary use of two modalities.

By default the tool provides some specific solutions in terms of possible CARE properties, which can be modified by the designers to suit their specific needs. Figure 1 shows the control panel to define the CARE properties that are made available or the refinement of the main abstract concepts (there is one tab for each of them). The CARE properties that have been deemed not meaningful appear greyed out. Designers can freely select those properties that seem more appropriate for their multimodal applications, and then the authoring environment will be able to generate user interfaces accordingly following transformations that will be introduced in the next sections. Thus, our environment allows the designers to customize what multimodal support to provide in user interface development.



Desktop Output

None [ ] Graphical Assignment [✓] Vocal Assignment [ ] Complementarity [✓] Redundancy [✓] Equivalence [ ]

Mobile Output

None [ ] Graphical Assignment [✓] Vocal Assignment [✓] Complementarity [✓] Redundancy [✓] Equivalence [ ]

Fig. 1. Control panel for customizing CARE properties.

While the CARE properties made available are similar for the two types of platforms that we consider (multimodal desktop and multimodal mobile), there are differences in the default properties proposed by the environment, taking into account the richer set of graphical resources of the desktop platform and that the mobile device can often be used on the move. Thus, in the case of the multimodal desktop, which has rich graphical resources, the composition operators are 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 case of a multimodal mobile, which has less rich graphical resources, the composition operators are supported both graphically and vocally, and the interaction elements are supported in such a way that the prompt is both vocal and graphical, the input either graphical or vocal, and the feedback is expressed in both modalities. The only-output elements can be both graphical and vocal or they use the two modalities in a complementary way, if they take a lot of resources.

Element type	**Interaction Phase**	**CARE Properties for Desktop**	**CARE properties for Mobile**
Composition Operator			
Grouping Relation	Output	**Graphical Assignment** Redundancy	Vocal Assignment Graphical Assignment **Redundancy**
Only Output Interactor			
Description, Object, Feedback, Alarm, Table	Output	**Graphical Assignment** Redundancy Complementarity	Vocal Assignment Graphical Assignment **Redundancy** Complementarity
Interaction Interactor			
Single/multiple selection Text Edit Numerical Edit	Input	Graphical Assignment **Equivalence**	Graphical Assignment **Equivalence** Vocal Assignment
	Prompt	**Graphical Assignment** Redundancy	Graphical Assignment **Redundancy** Vocal Assignment
	Feedback	**Graphical Assignment** Redundancy	Graphical Assignment **Redundancy** Vocal Assignment
Activator	Input	Graphical Assignment **Equivalence**	Graphical Assignment **Equivalence** Vocal Assignment
	Prompt	**Graphical Assignment** Redundancy	Graphical Assignment **Redundancy** Vocal Assignment
	Feedback	**Graphical Assignment** Redundancy	Graphical Assignment **Redundancy**
Navigator	Input	Graphical Assignment **Equivalence**	Graphical Assignment **Equivalence** Vocal Assignment
	Prompt	**Graphical Assignment** Redundancy	Graphical Assignment **Redundancy** Vocal Assignment
	Feedback	Vocal Assignment **None**	Vocal Assignment **None**

Table 2. How CARE Properties are made available for graphical+vocal desktop and mobile

Table 2 provides details on how the CARE properties are initially proposed by the environment to then generate graphical and vocal interfaces in both desktop and mobile platforms. Thus, it shows what properties have been deemed meaningful in the case of graphical and vocal interfaces, and these are made available in the authoring environment. We indicate in bold the specific properties that are initially pre-selected by default in the system. Thus, the properties in bold are those applied if the designer does not change anything in the tool. In particular, the first column indicates the element of the abstract interface considered. Different interaction phases (input, prompt, feedback) have to be considered depending on the interaction element in question.

In the case of only-output elements for the multimodal desktop platform the graphical assignment is proposed, while for the mobile one redundancy is suggested. For the interactive elements, in the desktop case we suggest equivalence for input and graphical assignment for prompt and feedback, while in the mobile case we prefer redundancy for prompt and feedback and still equivalence for input.

The composition operators aim to put together some interface elements in such a way that logical closeness or hierarchy of importance or some ordering is highlighted. 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).

The navigator allows the user to move from one presentation of the application to another. This type of element usually has no immediate feedback because the actual feedback is given by the change of the application presentation loaded. However, it is possible to have some kind of vocal feedback to indicate that a change of presentation is under way.

5 A Logical Language for MultiModality

In the MARIA framework the concrete languages are derived from the abstract one by refining the abstract vocabulary taking into account the considered platform and the associated interaction modality. In the case of a multimodal concrete language we have to consider refinements for multiple modalities and indicate how to compose them. In particular, the MARIA concrete language for composing graphical and vocal modalities is based on the two previously defined concrete languages (one for the graphical [13] and one for the vocal modality [14]). It adds the possibility to specify how to compose them through the CARE properties.

As we introduced before the MARIA abstract language structures a user interface in terms of a number of presentation. Each presentation has composition operators (usually groupings). The composition elements contain interactors that can be either interaction or only-output interface basic components, which can have events handlers associated to them indicating how they react to events. Each of these components of the language, ranging from the presentations to the elementary interactors, have different refinements for the graphical and the vocal modality and in the multimodal concrete language we indicate how to compose them. Thus, a multimodal presentation has associated both graphical settings (such as background colour or image or font settings) and vocal settings (such as speech recogniser or synthesis attributes). A grouping in the multimodal concrete language can exploit both visual aspects (using attributes such as position, dimension, border backgrounds) and vocal techniques (for example inserting keywords or sounds or pauses or changing synthesis properties). The interactors are enabled to exploit both graphical events (associated with mouse and keyboards) or vocal-specific events (such as no input or no match input or help request).

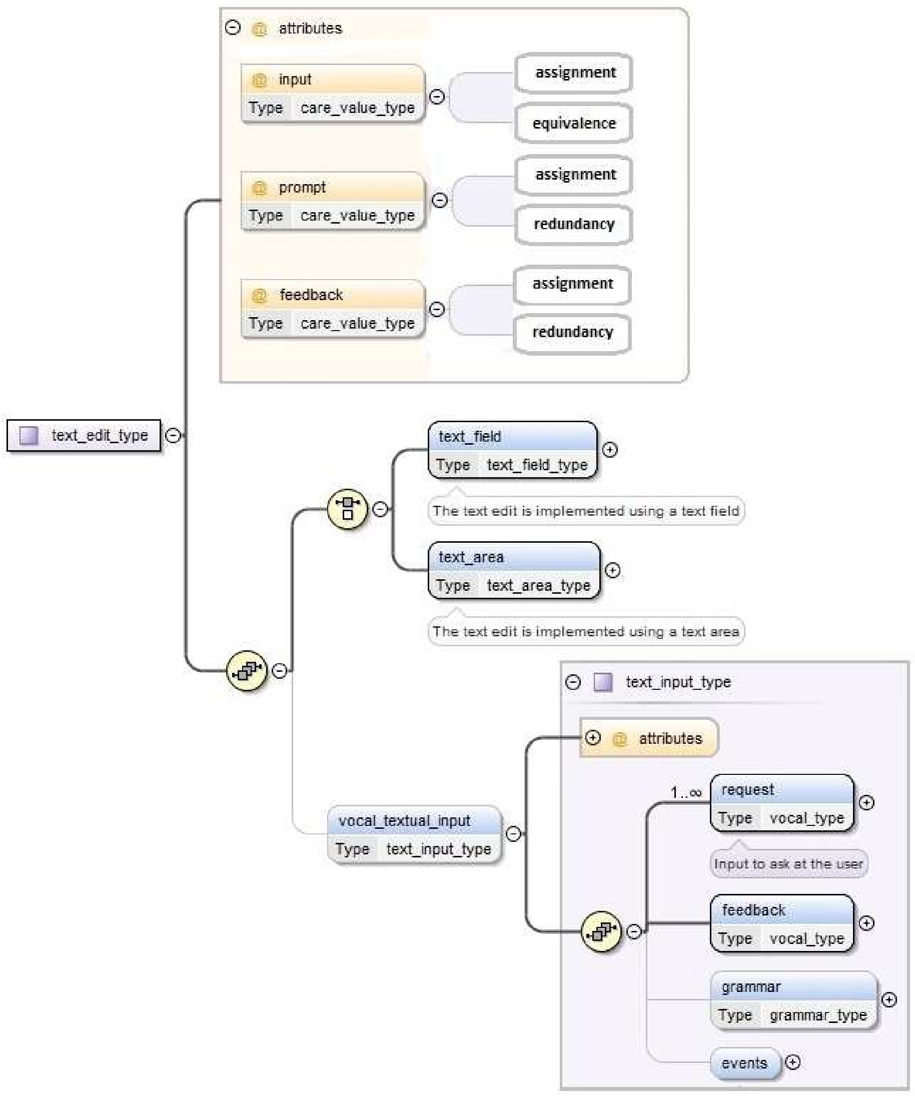


Fig. 2. An example of multimodal interactor derived from the graphical and vocal ones

In order to better understand how this approach works, we can take an example abstract interactor, the text edit. At the abstract level there is no assumption regarding the modality that should be used to perform this interaction. In Figure 2 there is a graphical representation of how this abstract interactor is refined into two parts depending on the modality, and then there are the possible CARE properties that have deemed meaningful for this interactor (in the top part of Figure 2). In the graphical case we have either a text area or a text field interactor as possible refinement, while in the vocal case we obtain a vocal textual input, which is composed of a request, a grammar to specify possible inputs and the associated feedback. Thus, the multimodal language includes both the vocal and the graphical refinements of the interactor, and adds attributes associated with instances of the CARE properties, which indicate the possible ways to compose them in the various interaction phases (input, prompt, feedback).

6 The Transformation into an Implementation

In terms of target implementation languages, we have considered X+V [1] 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. X+V is an integration of HTML and VoiceXML. The VoiceXML part is included in the head of the X+V document, while the HTML is in the body part. Thus, there is a clear distinction between these two parts in an X+V implementation. The connection between the two parts is obtained through the events and the associated handlers. For example, the expression:

```
<input type =" text " id =" from " name =" departure_city " ev: event ="
                    inputfocus " ev: handler ="# voice_city "/>
```

indicates that when the input focus event occurs in the from element of the graphical form then the voice_city event handler (which is managed in the vocal part) should be performed. In an X+V specification the synchronization between the values in the vocal and graphical part are obtained through the sync elements:

```
<xv: sync xv: input =" departure_city " xv: field ="# departure_city_field "/>
```

This sync element associates the value of an input element in the HTML part (departure_city) with the indicated field VoiceXML element (departure_city_field). This means that when an element is entered vocally then it is associated with both the vocal field and the input HTML element. The same result is obtained if the element is entered graphically. In addition, if the user changes the focus in the graphical part, then the corresponding vocal element, if any, is enabled. The sync element is not located in the VoiceXML form but it is a direct child of the HEAD element.

User interface generation is obtained through XSLT transformations [3]. They are obtained through stylesheets that transform an XML document into a new one in the target language (in our case the XML languages involved are the multimodal concrete MARIA language and X+V). The transformation is composed of a set template rules, which are defined by patterns indicating the source nodes conditions that should be verified, and templates indicating what corresponding element in the target document should be included. For example:

```
<xsl:template match=" c u i : p r e s e n t a t i o n ">
<html>
<head><t i t l e>Pr e s ent a t i on t i t l e</ t i t l e></head>
<body>Pr e s ent a t i on cont ent</body>
</html>
</xsl:template>
```

Indicates that a presentation in the source concrete language should be associated with the indicated elements in the corresponding HTML code.

The value of the CARE properties for the various user interface parts determines what should be generated. Assignment indicates whether only the vocal or only the graphical part is generated. Equivalence means that input in both modalities are generated, in particular for the vocal part a VoiceXML field is generated, for the HTML part an input element and then also a X+V element to synchronise the two parts. Complementarity and redundancy require generation of both the graphical and the vocal parts, even if they differ in the actual content that is generated.

The transformation is composed of three stylesheets: one for the graphical part and two for the vocal part, one to generate elements that are in already existing forms and one is for elements that require the creation of forms in which to put the currently generated element.

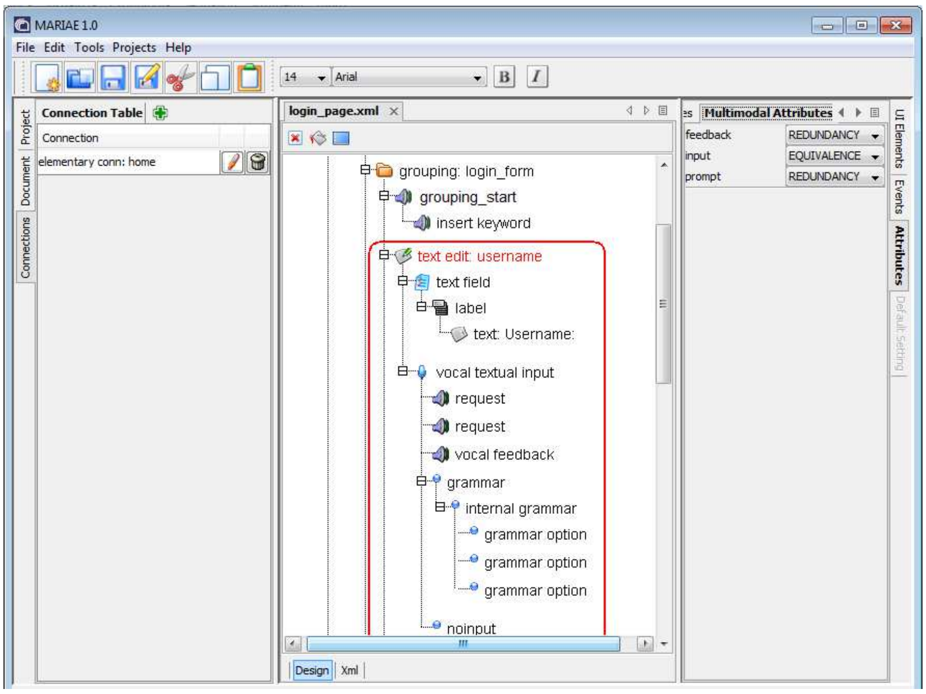
Thus, the transformation creates an X+V page for each presentation in the concrete description in such a way that in the head tag there is the call of the template to generate the X+V elements to synchronise the vocal and the graphical inputs and the templates to generate the vocal elements, while in the body tag there are the templates for generating the graphical elements. The X+V sync element is created only for the implementation of those interactors that are associated with the equivalence property for the input phase.

The transformation is also able to handle complex data structures such as tables. In the case tables must be rendered vocally, then it is possible to support either linear browsing (the elements are rendered line by line) or intelligent browsing, in which the corresponding header is rendered for each data element as well.

7 Authoring an Example Application

Tool support for the method presented has been implemented and integrated in the MARIAE environment, which is publicly available at http://giove.isti.cnr.it/tools/Mariae/. In order to see how it works we can consider an example application. We consider a home application, which allows users to control a number of domestic appliances.

The application is composed of four presentations: one for the user login, one showing the rooms that it is possible to monitor, one showing the appliances in the room selected, and one to change the settings of the appliance selected, if any.



feedback	REDUNDANCY
input	EQUIVALENCE
prompt	REDUNDANCY

```
grouping: login_form
  grouping_start
    insert keyword
  text edit: username
    text field
      label
        text: Username:
    vocal textual input
      request
      request
      vocal feedback
      grammar
        internal grammar
          grammar option
          grammar option
          grammar option
      noinput
```

Fig. 3. Authoring a multimodal concrete presentation.

Figure 3 shows the authoring environment in which the login presentation is being edited. The designer has specified a grouping element (*login_form*), which includes the input fields. It also contains a vocal element *grouping_start,* which is used to render a vocal message “Start login form!”. On the right-top part of the environment there is a panel for setting the multimodal attributes (the CARE properties) of the currently selected element. In the main central part there are the elements that compose the currently selected presentation. They are graphically represented as the XML syntax of the specification may be not easy to read and manage. The currently selected element highlighted in red is a text edit interactor for the entering of the user name. Since the CARE properties indicate the use of both graphical and vocal modality it has a graphical part with a text edit interactor and a vocal one with a vocal textual input interactor. The vocal part has two request elements with the count attribute, which allows developers to implement the tapered prompting technique. The first request asks for 'Insert your username'. In the case the user does not provide an input within a given time or the input is not recognised then the second request provides a more detailed indication of what has to be entered. The vocal textual input also allows the specification of a grammar for which the grammar options represent the possible inputs.

Figure 4 shows the multimodal implementation rendered through an Opera browser of the login presentation.

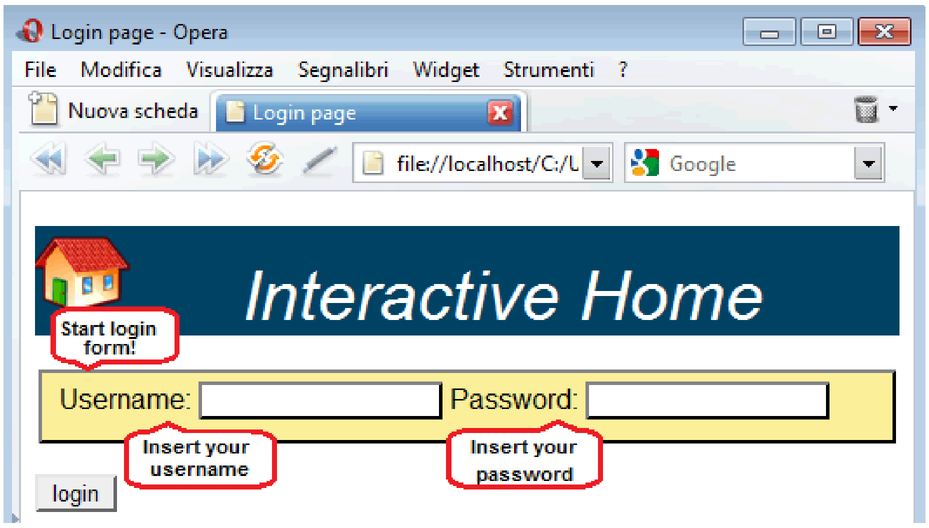
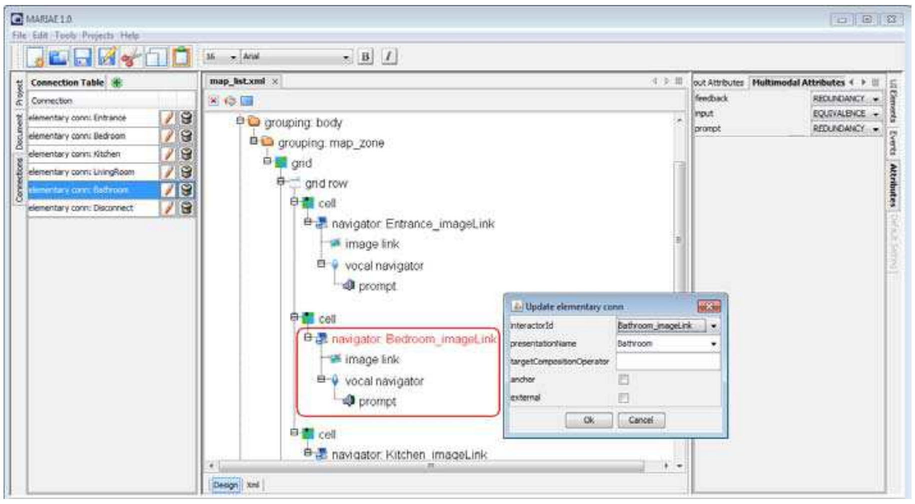


Fig. 4. The multimodal user interface corresponding to the previous presentation

Then, we can see (Figure 5) how it is possible to create connections among the various presentations through the authoring environment. The interactor_id attribute identifies the navigator interactor that triggers the presentation change, while presentation name indicates the target presentation. The Figure also shows the values of the multimodal attributes for such interactor (Feedback = Redundancy, Input = Equivalence, Prompt = Redundancy). By assigning such properties, which imply the full use of both graphical and vocal modalities, the navigator interactor includes a vocal part, with its prompt and feedback, and uses an image link for the graphical part.


MARIAE 1.0
File Edit Tools Projects Help
Project Document Connections
Design Xml
UI Elements Events Attributes

Connection Table

Connection
elementary conn: Entrance
elementary conn: Bedroom
elementary conn: Kitchen
elementary conn: LivingRoom
elementary conn: Bathroom
elementary conn: Disconnect

map_list.xml

```
grouping: body
  grouping: map_zone
    grid
      grid row
        cell
          navigator: Entrance_imageLink
            image link
            vocal navigator
              prompt
        cell
          navigator: Bedroom_imageLink
            image link
            vocal navigator
              prompt
        cell
          navigator: Kitchen_imageLink
```

Multimodal Attributes

feedback	REDUNDANCY
input	EQUIVALENCE
prompt	REDUNDANCY

Update elementary conn

interactorId	Bathroom_imageLink
presentationName	Bathroom
targetCompositionOperator	
anchor	☐
external	☐

Ok Cancel

Fig. 5. Editing connections among multimodal presentations.

Once the new presentation has been completed we obtain a presentation for the room selection. It contains a grouping with an initial vocal message 'Select the room you want to monitor' to introduce the navigator elements associated with each selectable room. For each navigator there is a vocal prompt that indicates what vocal input to enter to select the corresponding room (e.g. 'Say living to go to the living room'). Figure 6 shows the corresponding user interface implementation.

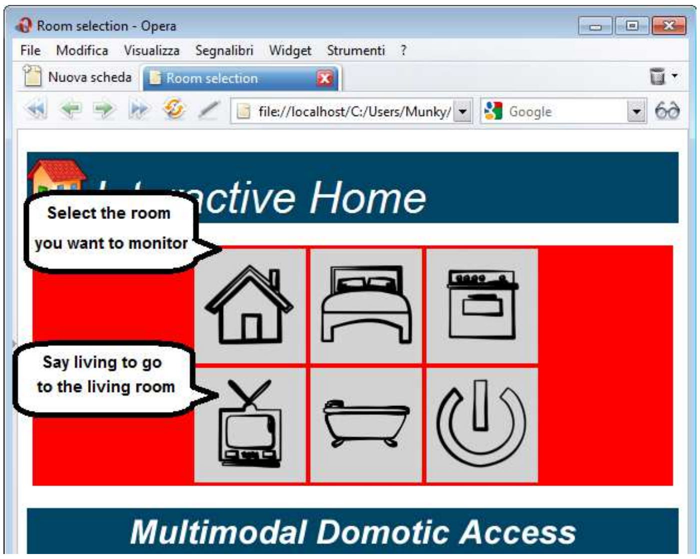

Room selection - Opera
File Modifica Visualizza Segnalibri Widget Strumenti ?
Nuova scheda
Room selection
file://localhost/C:/Users/Munky/
Google
Interactive Home
Select the room you want to monitor
Say living to go to the living room
Multimodal Domotic Access

Fig. 6. The multimodal user interface implementation supporting the multiple connections.

Conclusions and Future Work

This work introduces a novel logical language for multimodal interfaces and the associated environment, which allows designers to easily compose multimodal interfaces and derive X+V implementations. It provides designers with the possibility to work through logical descriptions of the user interface and support for choosing the most suitable combination of various modalities at different granularity levels and for the various parts of the user interface.
This has been integrated in an environment for multi-device interface design and development, thus facilitating the implementation of multiple versions adapted to the various target modalities because of the use of a common abstract vocabulary, which is then refined according to the target platforms. This avoids requiring developers to learn a plethora of details of the many possible implementation languages
This result has been validated through the development of some multimodal applications (one of them is briefly described in the paper), which can be rendered through publicly available browsers (Opera). The authoring environment is publicly available for download of the executable code.
Future work will be dedicated to empirical tests in order to better assess how the development process is facilitated with this approach, especially when multi-device interfaces should be developed (e.g. desktop, mobile, vocal and multimodal versions of the same application).
We also plan to develop an automatic system able to support graphical-to-multimodal user interface content adaptation. Future work will be also 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.

Acknowledgments

This work has been supported by the EU ICT STREP Project ServFace (http://www.servface.eu/)

References

1. Axelsson J., Cross C., Lie H.W., McCobb G., Raman T.V., and Wilson L.. XHTML+Voice Profile 1.0. Recommendation, World Wide Web Consortium (W3C), 2001. See http://www.w3.org/TR/xhtml+
2. Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, O., Marucci, L., Paternò, F., Santoro, C., Souchon, N., Thevenin, D., Vanderdonckt, J.: The CAMELEON reference framework. CAMELEON project, Deliverable 1.1 (2002).
3. Clark J., Xsl Transformations (XSLT) version 1.0. Technical report,W3C, 1999.
4. Coutaz J., Nigay L., Salber D.,.Blandford A, May J., Young R., 1995. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE Properties. Proceedings INTERACT 1995, pp.115-120.

5. Honkala M., Pohja M.: Multimodal interaction with XForms. Proceedings ICWE 2006: 201-208.
6. Lawson J., Al-Akkad A., Vanderdonckt J., Macq B.: An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components. Proceedings ACM EICS 2009: 245-254
7. Lin, J., Landay, J.A.: Employing Patterns and Layers for Early-Stage Design and Prototyping of Cross-Device User Interfaces. Proc. CHI: 1313-1322 (2008)
8. Myers, B.A., Hudson, S.E., Pausch, R.: Past, Present and Future of User Interface Software tools. ACM Trans. Comput. Hum. Interact. 7, 3–28 (2000)
9. Multimodal Interaction Activity (W3C), http://www.w3.org/2002/mmi/
10. 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.
11. Obrenovic, Z., Starcevic D., Selic B., A Model-Driven Approach to Content Repurposing, IEEE Multimedia, January March 2004, pp.62-71.
12. Paternò, F., Giammarino F.: Authoring interfaces with combined use of graphics and voice for both stationary and mobile devices. AVI 2006: 329-335
13. Paternò, Santoro, C., Spano, L.D.: MARIA: A Universal Language for Service-Oriented Applications in Ubiquitous Environment. ACM Transactions on Computer-Human Interaction, Vol.16, N.4, November, pp.19:1-19:30 (2009)
14. Paternò F., Sisti C., Deriving Vocal Interfaces from Logical Descriptions in Multi-Device Authoring Environments, Proceedings ICWE 2010, Wien, July 2010, Springer Verlag, LNCS 6189, pp.204-217.
15. Stanciulescu, A., Limbourg, Q., Vanderdonckt, J., Michotte, B., Montero, F.: A Transformational Approach for Multimodal Web User Interfaces based on UsiXML. Proc. ICMI: 259-266 (2005)