Chapter 4
Extending MARIA to Support Distributed User Interfaces

Marco Manca and Fabio Paternò

Abstract In this paper, we describe a solution to obtain flexible user interface distribution across multiple devices, even supporting different modalities. For this purpose we extend a model-based language and consider various user interface granularities. We also explain how this solution works at run-time in order to support dynamic distribution of user interface elements across various devices.

[AU1] ## 4.1 Introduction

The current technological trends are determining a steadily increasing number of computers per person along with many sensors able to detect a wide variety of contextual events. The computers are becoming more and more variegated in terms of possible interaction resources and modalities, including interconnected embedded devices composed of small electronic components, which can interact with each other.

This implies that in the near future we will no longer access our applications through one device at a given time but we will rather use sets of collaborating devices available while moving, such as using the smartphone to control the content on a large screen. Thus, emerging ubiquitous environments need Distributed User Interfaces (DUIs), which are interfaces whose different parts can be distributed in time and space on different monitors, devices, and computing platforms, depending on several parameters expressing the context of use [3]. This has an impact on the user interface languages and technologies because they should be able to support the main concepts charactersing interactions with an application through various combinations of multiple devices.

[AU2] M. Manca (✉) • F. Paternò
CNR-ISTI, HIIS Laboratory, Via Moruzzi 1, 56124 Pisa, Italy
e-mail: Marco.Manca@isti.cnr.it; Fabio.Paterno@isti.cnr.it

J.A. Gallud et al. (eds.), *Distributed User Interfaces: Designing Interfaces for the Distributed Ecosystem*, Human-Computer Interaction Series,
DOI 10.1007/978-1-4471-2271-5_4,

Model-based approaches have been considered in order to manage the increasing complexity derived from managing user interfaces in multi-device environments, since each device has specific interaction resources and implementation languages to execute such user interfaces. They are also currently under consideration for W3C for standardization purposes [4]. The basic idea is to provide a universal small conceptual vocabulary to support user interface design, which can then be refined into a variety of implementation languages with the support of automatic transformations without requiring developers to learn all the details of such implementation languages.

Some research effort to address distributed user interfaces with model-based approaches has already been carried out but with limited results and not able to support the many possible ways to distribute user interface elements. In HLUID (High Level UI Description) [8] the user interface has a hierarchical structure and the leaves of the tree are Abstract Interactor Object (AIOs) describing high-level interactors. During the rendering process the AIOs are mapped onto Concrete Interaction Object (CIOs) associated with the current platform. In addition, they introduce a split concept for the groupings through an attribute that, when it is set to true, allows the distribution of the user interface elements without losing its logical structure. In our case we propose a different solution, still using a model-based approach. One difference is that we support the specification at the concrete level because at this level it is easier to generate the corresponding implementations and there is a better understanding of the actual effects that can be obtained. Vanderdonckt and others [6] have developed a set of primitives to manage user interface distribution but they only consider graphical user interfaces while our approach is able to support user interfaces exploiting also other modalities, such as voice. Blumendorf and others [1] address multimodal interfaces but they lack an underlying language able to support user interface distribution.

To overcome the limitations of previous work our starting point is the MARIA language [7], which in current version consists in a set of languages: one for abstract user interface description, and a set of concrete refinements of such language for various target platforms (Vocal, Desktop, Smartphone with touch, Mobile, Multimodal desktop, Multimodal mobile). Then user interfaces generators for various implementation languages (XHTML, SMIL, VoiceXML, X+V, HTML 5) are available starting with such concrete languages. Tools for authoring user interfaces in MARIA and for reverse engineering Web pages into MARIA specifications are publicly available at http://giove.isti.cnr.it/Tools/

We have extended such language in order to be able to specify distributed user interfaces and we have also designed a solution to generate implementations of such distributed user interfaces, which can dynamically change how the user interface elements are distributed according to user requests or other events.

In the paper we first provide an overview of the solution that we have developed. Next, we provide some detail on the language supporting it and show some example application.

Lastly, we draw some conclusions and provide indications for future work.

4.2 The Approach

The approach proposed has been developed aiming two satisfy two main requirements:

- flexible support able to address a wide variety of granularities in terms of user interface components to distribute;
- small and simple set of primitives to indicate how to perform the distribution.

Regarding the set of primitives we decided to use the CARE (Complementarity, Assignment, Redundancy, and Equivalence) properties [2], which were introduced to describe multimodal user interfaces, and have already been considered in the MARIA concrete language for multimodal interfaces [5]. In our case the idea is to use them with this meaning:

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

Regarding the possible granularity levels to address we have started from the consideration that in MARIA a user interface is composed of presentations (in graphical interfaces they correspond to the set of elements that can be perceived at a given time, e.g. a Web page). Then, in each presentation there can be a combination of user interface elements and instances of composition operators. In MARIA there are three types of composition operators: grouping (a set of elements logically related to each other), relation, a relation between groups of elements (e.g. in a form there usually are a set of interactive elements and a set of associated control elements to send or clear them), repeater (a group of elements that are repeated multiple times). Since we aim to obtain full control on what can be distributed we decided to consider also the possibility of distributing the elements within a single interaction element. For example, a text edit interactor can be distributed in such a way that the user enter the text in one device but receives feedback on what has actually been entered in another device. For this purpose we provide the possibility to decompose interactive interface elements into three subparts: prompt, input, and feedback.

Then, by combining the set of four possible granularity levels (presentations, compositions, interface elements, interactive subparts) with the CARE properties we obtain a simple and powerful tool to indicate how the user interface can be distributed in a flexible way. Thus, we can distribute an entire presentation. For example, by associating the Redundancy property we indicate that one presentation should be completely rendered in two different devices. However, we can also distribute single

interface elements. For example, distributing a textual object in a complementary way means that part of the text is rendered through one device and part through another one. As we mentioned, it is even possible to distribute sub-elements of a single interaction object. For example, a single selection object can be distributed in such a way that when the user selects one element then the feedback indicating what has been selected is rendered through another device. This means that the prompt and input components have been assigned to one device while the feedback sub-component to another one.

It is worth pointing out that the decomposition into prompt, input and feedback is meaningful only for interactive interface element, and cannot be applied for only-output elements.

In this way it is also possible to easily indicate how dynamically the user interface elements can be distributed. Thus, if we want to move one element from one device to another then it means that we have changed the device to which that element is assigned. While if we want to copy a user interface element from one device to another then it means that we have changed the corresponding CARE property from Assignment to Redundancy.

4.3 The Language

In order to formalise the concepts introduced in a language that can be used to design and generate the corresponding user interfaces we have extended the MARIA language.

In particular, we have introduced a language with the possibility of defining a concrete distributed user interface. In such language it is possible to indicate the types of devices on which the user interface can be distributed. Each device belongs to a platform for which already exists a corresponding concrete language. Such concrete languages refine the abstract vocabulary taking into account the interaction resources that characterise the corresponding platform. This allows designers to specify interface elements that better adapt to the devices in which they are rendered.

The user interface is hierarchically structured: the user interface is composed of presentations. Each presentation is composed of a combination of interface elements and composition elements, which can be recursively composed of interface and composition elements. When a CARE property is associated to one element of this hierarchy then all the underlying elements inherit such association. Thus, if a grouping of elements is assigned to a device then all the user interface elements of the group will be assigned to it. This also simplifies the specification process by avoiding the need to indicate the value of the CARE properties to all the interface elements.

Below we can see an excerpt from an example of MARIA specification of a distributed user interface. We consider a grouping of interactor elements. The corresponding CARE property is complementarity, which means that the interface elements are distributed across various devices. In particular, there are two interactors (a video and a text) and four devices. For each device it is specified the corresponding platform, in this case we have one desktop, one vocal, one mobile, and one multimodal.

```
<grouping>
    < care value="complementarity">
        <bind>
            <interactor interactor_id="description_video"/>
            <device id="paterno" platform="desktop"/>
        </bind>
        <bind>
            <interactor interactor_id="description_text"/>
            <device id="sisti" platform="vocal"/>
            <device id="iphone_lab" platform="mobile"/>
            <device id="manca" platform="multimodal"/>
        </bind>
    </ care>
```

Afterwards we have the specification of the two involved interactors. Since the text is redundant over three devices, we do not have to specify again the CARE attributes value, which is inherited from the parent grouping. Actually, since in one case one device is multimodal, we can again apply the CARE properties to indicate how the information is distributed across the two modalities of the same device. In the example, it is redundant again.

```
<description id="description_video">
    <description_desktop>
        <video src="video.flv" alt="alternative_text"/>
    </description_desktop>
</description>
<description id="description_text">
  <!—REDUNDANT DISTRIBUTION -->
  <description_mobile>
      <text><string> Text </string></text>
  </description_mobile>
  <description_vocal>
      <speech><content>Text </content></speech>
  </description_vocal>
  <description_multimodal output="redudancy">
      <!-- [graphical part] -->
      <text><string> Text</string></text>
      <!-- [vocal part] -->
      <speech><content> Text</content></speech>
  </description_multimodal>
</description>
</grouping>
```

A dynamic change of the distribution of the user interface elements is supported by adding a distribution event in the dialogue model in the MARIA specification. The dialogue model is composed of a number of event handlers and indicate the temporal relation among them. The distribution event can be triggered by a user action and the event handler indicates what user interface elements and what devices are involved by changing the corresponding CARE attributes.

In the following we can see an example of such events. It is generated by a button that when pressed activates a distribution of the input, prompt, and feedback components

of one interactor in such a way that the input can be entered either though a desktop or a vocal device, the prompt is complementary across such two devices, and the feedback is assigned to only the desktop device.

```
<activator>
    <button><label>distribute UI</label><button>
    <event>
        <distribution_event>
            <handler>
                <change_property interactor_id=”multiple_id”
phase=”input” care_value=”EQUIVALENT”>
                    <devices id=”manca_pc” platform=”desktop”/>
                    <devices id=”sisti_pc” platform=”vocal”>
                </change_property>
                <change_property interactor_id=”multiple_id”
phase=”prompt” care_value=”COMPLEMENTARITY”>
                    <devices id=”manca_pc” platform=”desktop”/>
                    <devices id=”sisti_pc” platform=”vocal”>
                </change_property>
                 <change_property interactor_id=”multiple_id”
phase=”feedback” care_value=”ASSIGNMENT”>
                     <devices id=”manca_pc” platform=”desktop”/>
                 </change_property>
            </handler>
        </distribution_event>
    </event>
</activator>
```

4.4 An Example Application

In order to show how our approach works let us consider a concrete example, not too complex for sake of clarity. We consider an application to show slides, it allow users to go back and forth, and to annotate them. Figure 4.1 shows the initial user interface, completely rendered in a desktop graphical device.

More precisely Fig. 4.2 shows the corresponding hierarchical structure: one presentation with a couple of output descriptive objects (the application title and the slide), a grouping composing two buttons to go back and forth, an interactive elements to write comments and a button to store them. Since the interface is completely shown in one single device, it is sufficient to associate the assignment property to the root.

Now, suppose that the user wants to distribute parts of the user interface to a multimodal mobile device as indicated in Fig. 4.3.

To obtain this example of distributed user interface a number of elements have been assigned new values of the CARE attribute as indicated by Fig. 4.4. Thus, there is no longer a CARE attribute assigned at the presentation level. The title description is redundant while the slide description is assigned to the desktop device, because it has a larger screen that can better show its content. The grouping with the buttons for going forth and back is assigned to the mobile device and it has a multimodal support: prompt is redundant with vocal and graphical modality, and input is equivalent

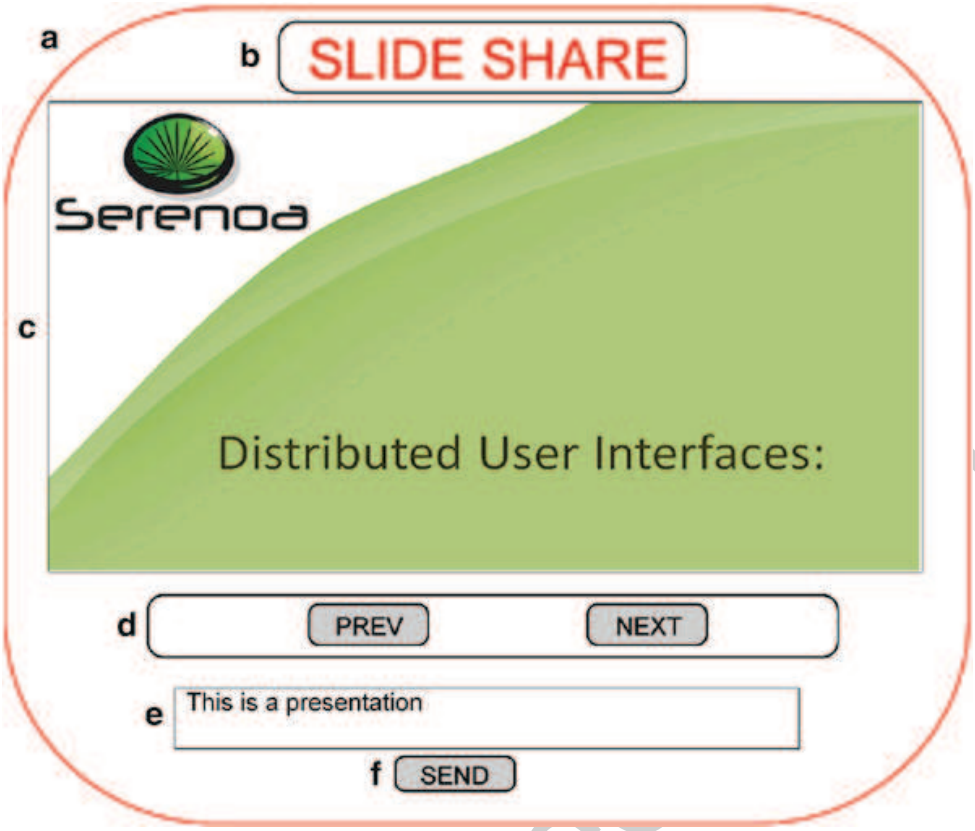

Fig. 4.1 The slide share application

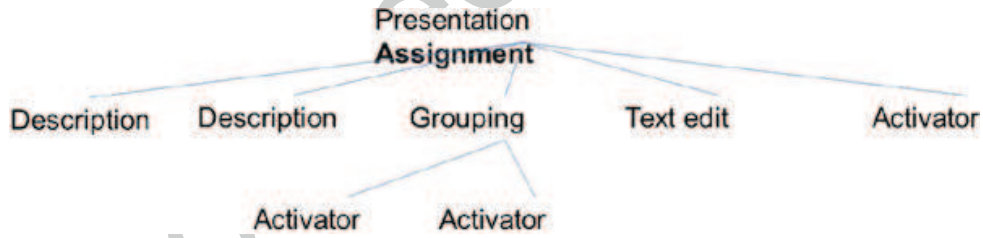

Fig. 4.2 The structure of the example

and can be provided by either modality. The text edit interactor for entering comments is redundant in both devices but the button to store the comments is assigned only to the mobile device, for immediate activation by the user.

4.5 Conclusions and Future Work

Distributed user interfaces require novel languages and tools in order to obtain flexible support. In this paper, we have presented an approach able to describe distribution at various granularity levels, even involving multimodal devices.

Future work will be dedicated to the design and development of a software architecture supporting the corresponding implementation.

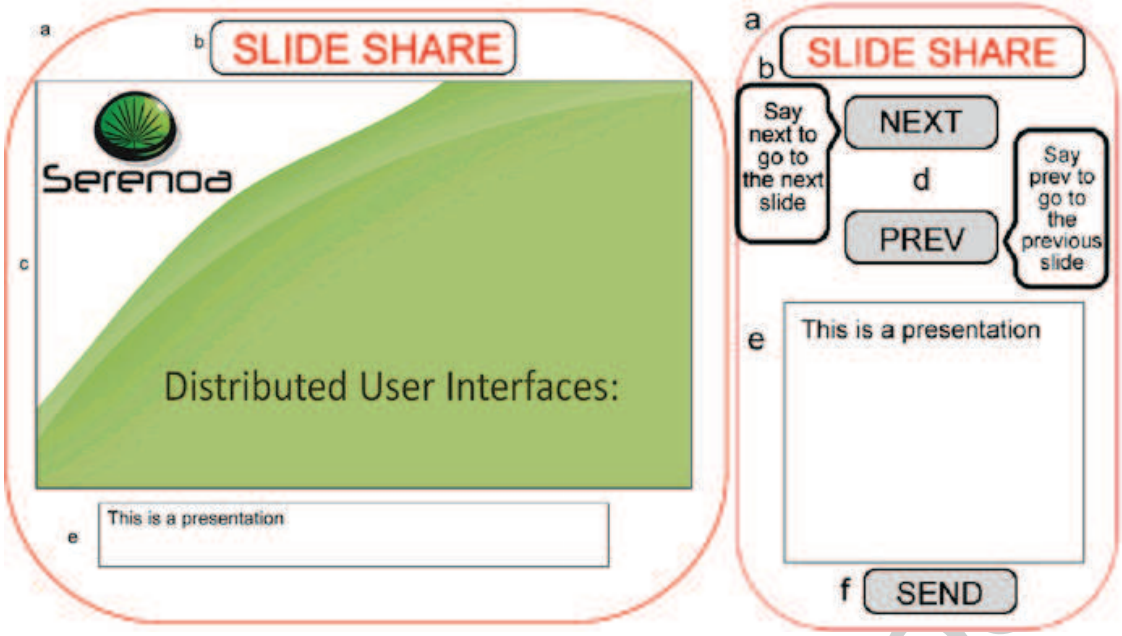

Fig. 4.3 The slide share distributed

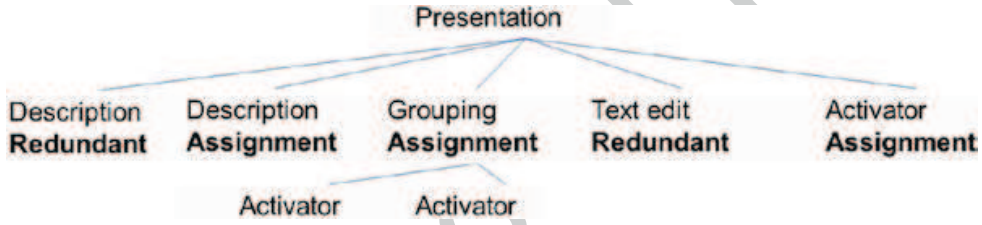

Fig. 4.4 The updated CARE attributes

References

1. Blumendorf, M., Roscher, D., Albayrak, S.: Dynamic user interface distribution for flexible multimodal interaction. In: Proceedings ICMI-MLMI'10, Beijing, pp. 8–12 (2010)
2. Coutaz J., Nigay L., Salber D., Blandford A, May J., Young, R.: Four easy pieces for assessing the usability of multimodal interaction: The CARE properties. In: Proceedings INTERACT 1995, Lillehammer, pp. 115–120 (1995)
3. Demeure, A., Sottet, J.S., Calvary, G., Coutaz, J., Ganneau, V., Vanderdonckt, J.: The 4C reference model for distributed user interfaces. In: Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems, Washington, DC, pp. 61–69 (2008)
4. Fonseca, J.M.C. (ed.): W3C Model-based UI XG Final Report, May 2010. http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/
5. Manca, M., Paternò F.: Supporting multimodality in service-oriented model-based development environments. In: Proceedings HCSE 2010, 3rd Conference on Human-Centred Software Engineering, LNCS 6409, pp. 135–148. Springer, Reykjavik (2010)
6. Melchior, J., Vanderdonckt, J.: A model-based approach for distributed user interfaces. In: Proceedings ACM EICS 2011, Pisa (2011)
7. Paternò, F., Santoro, C., Spano, L.D.: MARIA: a universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Comput. Hum. Interact. **16**(4) (2009)
8. Vandervelpen, C., Conix, K.: Towards model-based design support for distributed user interfaces. In: NordiCHI 2004, Tampere, pp. 61–70 (2004)