# Customizable Dynamic User Interface Distribution

**Marco Manca, Fabio Paternò**
CNR-ISTI, HIIS Laboratory
Pisa, Italy
{marco.manca, fabio.paterno}@isti.cnr.it

**ABSTRACT**

This paper describes a solution for flexibly obtaining distributed user interfaces across multiple devices. To this end, we propose a model-based approach, with associated authoring environment, which allows designers and developers to specify how to distribute interfaces at various granularity levels, ranging from entire user interfaces to parts of single interactive elements, and obtain the corresponding implementations. This solution includes run-time support for keeping the resulting user interfaces synchronized and customization tools that allow end users to dynamically change how the user interface elements are distributed across multiple interactive devices in order to address unforeseen situations. We also report on a first user test and how the environment has evolved according to the user feedback.

**Author Keywords**

Distributed User Interfaces, Multi-device User Interfaces, Model-based User Interface Description Languages.

**ACM Classification Keywords**

H.5 Information Interfaces and Presentation; H.5.2 User Interfaces.

## INTRODUCTION

The number of computers per person has been increasing steadily to the point where users may have even more than five devices [3]. Enabling seamless interaction across multiple devices would enhance the users' flexibility. The way designers address multi-device user interfaces is through responsive design [11] in which they define different versions of the user interface mainly depending on the available screen size. However, this is not sufficient. How people access their interactive applications will change, with users no longer accessing their applications through one device at a given time, but rather using multiple collaborating devices available while mobile, such as using the smartphone to control the content on a large screen. The technological trends towards ubiquitous environments require Distributed User Interfaces (DUIs), which are user interfaces that allow users to access concurrently an interactive application through multiple devices at a given time.

Current user interface languages and technologies have not been designed to support concurrent access from multiple devices, and therefore need to be enhanced in order to be able to handle the main concepts characterising interactions with an application through various combinations of multiple devices. Indeed, such novel user interfaces require that parts of the interfaces on different devices can be kept synchronised according to user input or that some parts can be dynamically migrated from one device to another through both push and pull mechanisms while preserving their state. While some applications exploiting distributed user interfaces have been proposed (for example [14]), such applications have usually applied specific ad hoc solutions, which are difficult to generalize. Thus, there is a need for solutions able to provide support for distributed user interfaces for various applications, especially if the distribution of the user interface elements needs to be changed dynamically.

Model-based approaches [5] have been proposed to manage the increasing complexity derived from user interfaces in multi-device environments. They have also been considered in W3C for standardization in order to ease their industrial adoption1. The main idea is to provide a small general conceptual vocabulary to support user interface developers in the design process. The resulting logical descriptions can then be transferred into a variety of implementation languages with the support of automatic transformations, thereby hiding the complexity deriving from the heterogeneous languages and devices, and sparing developers the need to learn all the details of such implementation languages. Thus, they can be useful to obtain more accessible applications as well since they make more explicit the semantics and the role of the various user interface elements, which is also important for access through assistive technologies. In the area of distributed user interfaces, model-based approaches can determine more general solutions that involve different interaction modalities or implementation environments since they provide general frameworks that can be instantiated across them.


EICS'16, June 21-24, 2016, Brussels, Belgium
 ISBN 978-1-4503-4322-0/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2933242.2933259

[1] http://www.w3.org/TR/2014/NOTE-mbui-intro-20140107/

## RELATED WORK

Multibrowsing [8] was an early framework that extended the browsing metaphor across multiple displays. It provided support to bring up Web pages on any of the shared screens. In this environment the enhanced clients needed a custom browser plug-in and the target displays had to run a special service to receive the distribution event and open the pages. In addition, there was only one available command (browse(URL)), applied to an entire user interface. iRos [17] was its middleware platform used to connect devices that have their own low-level operating system. There could only be two types of distribution in this environment: one to one and one to many. Our approach is more general, supporting more flexible distributions as well, and does not need any plug-in to distribute a Web page. Moreover, we allow users to distribute also subparts of the interfaces.

Penalver et al. [16] provided useful indications regarding important aspects in DUIs: the UI as a whole or its elements should be easily transferred across platforms and devices; DUI systems should be decomposable meaning that if a given UI is composed of a number of elements, one or more of these elements can be executed independently without losing functionality; different elements of the same DUI should be manageable at the same time in different platforms. However, they do not provide supporting software architectures. Melchior et al. [13] developed an approach by which they describe the distribution in terms of 'Distribution Primitives', a set of 11 basic operations for supporting distribution (display, undisplay, copy, move, etc.). We opted to use CARE (Complementarity, Assignment, Redundancy, Equivalence) [1] properties for this purpose because they provide a more compact vocabulary still able to express the relevant concepts (their meaning and use is detailed in the next section). In addition, our approach is able to support dynamic user interfaces through customization tools that can run on various devices (including smartphones) and support distribution at a broader set of granularities levels.

Felix et al. [4] developed a system to share information for synchronous collaboration. They provide a shared workspace to exchange documents between users participating in the session. The Drag&Share system allows documents to be distributed, but not the UI elements, even though it uses a system architecture with some similarities to our own. Lorenz et al. [9] developed a prototype that consists of interactive components distributed amongst two devices: a handheld and a remote device for controlling an application on a remote screen. However, they defined this distribution statically and it was not possible to modify how to distribute the interactive application at run-time. Rachid et al. [18] investigated solutions for distributing user interfaces in order to assess their usability in performing a set of tasks. This type of investigation can be useful to understand what distributions are more suitable for the various task types, but does not identify the underlying support that should make them.

Some research effort has addressed distributed user interfaces with model-based approaches, but with limited results and a lack of support for the many ways to distribute user interface elements. [10] showed how an interactive system can be distributed among several devices utilizing UIML to create interface elements and each UI element is related to a task. Users manually indicate the UI elements that are displayed on each device but they cannot decompose the UI into different parts and distribute them in different ways. There are no commands or definitions such as the CARE properties. Blumendorf et al. [2] address multimodal interfaces, but without an underlying language able to specify user interface distribution in general terms. Martinie et al. [12] propose a model-based approach for the design and development of distributed user interfaces in the context of safety critical systems. The new distributed user interfaces can be generated at runtime to provide a new user interface organizing the information required to perform the task; thus they do not provide a flexible way for the end users to choose which interface element to distribute, but the distribution is driven by tasks and procedures associated to the UI.

Frosini et al. [6] have proposed a framework and an associated run-time environment that supports distribution across dynamic and multi-user environments. They provide only run-time support that saves the distribution state and a library that is used by developers to introduce the UI distribution in their applications. However, there is no support for users to customize the possible distributions in ways different from those originally indicated.

Panelrama [19] aims to support distributed user interfaces by organizing the HTML code in a flat panel hierarchy: each panel groups a number of elements and is associated with some properties (such as size, touch capability, proximity to users) so that when relevant devices are detected nearby then distribution is automatically triggered by moving the panels to the devices that best fit the corresponding properties. Thus, Panelrama does not support the various granularity levels of our approach nor distribution customization by users.

The MultiMasher tool [7] is a mashup environment able to support push of some components on multiple devices. Thus, it allows developers to design only applications that can be obtained from components of the existing Web applications that the environment is able to manage, and, as with all mashup environments, it is not able to support development of new applications from scratch. To summarize, there is still a lack of general solutions able at the same time: to facilitate obtaining implementations for different devices, as happens with model-based approaches; to flexibly specify DUIs, by providing a simple set of properties, which can be applied to a wide variety of granularities, to allow end users to dynamically customize the distribution across different types of platforms to better address unforeseen circumstances.

## CONCEPTUAL DESIGN OF THE PROPOSED SOLUTION

In this section we provide an overview of the proposed solution and introduce how distribution is modelled, we also indicate how the various parts of the solution are detailed throughout the paper.

### Solution Overview

This solution allows developers and end users to flexibly obtain distributed user interfaces across multiple devices. This means that how distribution should occur can be specified at design time, and further modified during actual use. Figure 1 provides an overview of the main elements of the solution: at design time it is possible to edit the distributed user interfaces through a specific authoring tool exploiting an extended version of a model-based language (Distributed MARIA) using a small set of properties (the CARE properties) that allow designers to specify the possible distribution of the various parts of the user interface. The advantage of the model-based solution is not only the possibility of obtaining consistent versions in various devices. It opens up the possibility of a common abstract language that identifies the semantics of the possible user interface elements, and then obtaining various concrete refinements for various platforms. The proposed solution is general in various perspectives: it is not hardwired for a specific implementation language or application; it supports the possibility of customizing the distribution in various phases: design, execution, use; and it is able to support user interface distribution in a wide range of granularities, thus supporting flexibility in general terms.

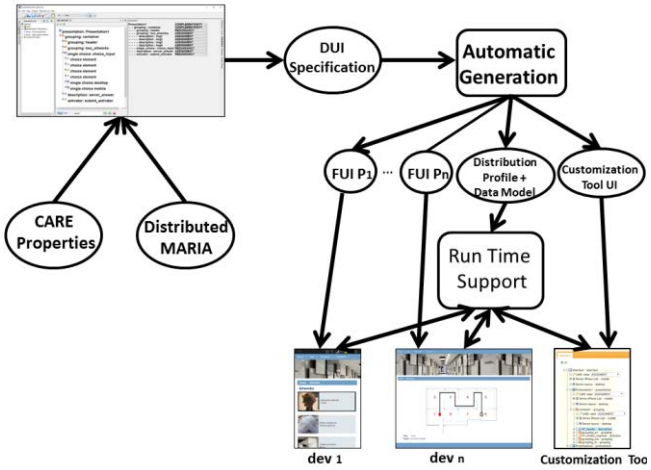


**Figure 1. The Proposed Solution for Customizable Distributed User Interfaces**

The resulting specification is provided as input for automatic transformation that generates the implementation of the corresponding final user interfaces (FUI in the Figure 1) for the various types of devices, the corresponding distribution profile (composed of the CARE values and the corresponding devices for each UI element) along with the data model (description of all data handled by the interface) that will be used to keep the state of the user interface, and customization support. Then, there is run-time support that is able to keep synchronized the various distributed user interface parts during their use by exploiting such information, and also allows users to change the distribution configuration in order to meet unplanned needs through customization tools.

### How Distribution is Modelled

Our work provides flexibility in the possible user interface distributions that can be obtained. One important aspect is to support the choice of a range of granularity levels of the elements to distribute across the various devices. Thus, we provide the possibility of distributing elements at the following granularity levels: the entire application user interface; an entire presentation; compositions of elements (parts of the user interface identified by composition operators); elementary interactors (interactive or output-only elements); parts of single interactive interactors, for single interactive interactions it is possible to distinguish three subparts: *prompt*, *input*, *feedback*; where a prompt is the part of the interactive element that indicates that the system is ready to receive the corresponding input (e.g. a label explaining the type of input expected); input is the part dedicated to receive some information from the user; and feedback is the response obtained through the interactive element after an input (for example, it may correspond to a checkmark when a user clicks on a checkbox or to a text in a field after a keyboard user input). In this way we provide control over distribution of even the most elementary aspects of a user interface, thereby allowing for example the insertion of input only to a single device, while the others receive the feedback of this action. Decomposition into prompt, input and feedback is meaningful only for interactive elementary interactors, and cannot be applied for output-only interactors and composition operators.

A user interface can be logically represented through a tree-like hierarchical structure based on this classification of its component elements.

In order to specify how to distribute such elements we use the CARE properties. Originally introduced in [1], we have interpreted the general properties in a specific manner more suitable for the current work: *Complementarity*, it indicates that a part of the element can be distributed onto one device and the remaining parts onto the other(s); for example, distributing a grouping composition in a complementary way means that some elements contained therein are distributed on some devices and the remainder on others; *Assignment*, the element is assigned to only one device; *Redundancy*, the element is duplicated on multiple devices; *Equivalence*, the user interface element can be supported by either one device or another.

To describe how to distribute the interface, designers have to indicate the CARE properties for the user interface elements to distribute, and the specific devices or the platforms that should be exploited for this purpose. We

chose such CARE properties because they represent a simple vocabulary for describing various ways to distribute user interface elements. Indeed, the CARE properties together with the hierarchical structure of the user interface description, facilitate the flexible specification of the distribution. When the properties are associated to an element that is high in the hierarchy representing the interactive application structure, then they are inherited by all child elements. Thus, if an interface or a presentation is assigned to one device, then all elements that it contains are assigned to that device as well. This applies to all the intermediate elements in the interface structure hierarchy. This kind of inheritance provides a simpler and less verbose definition of the distribution by defining it only in the intermediate elements in the hierarchy.

Another example can be a user who enters an input from a smartphone and this value is shown in a public display as well. This is possible by associating the CARE properties with the three different subparts of the interactive interactor (input, prompt, and feedback): prompt and input subparts are both associated with the assignment property value to the smartphone, while the feedback subpart is assigned to the redundancy property on both smartphone and public display. The concepts just introduced enable dynamically changing the distribution of the user interface elements across different devices. For example, a set of elements assigned to one device can be moved to another one by simply changing the device associated with the assignment property. Instead, if the desired operation is to copy an element to another device, it is sufficient to change the value of the corresponding CARE property from Assignment to Redundancy and add the new device identifier to the list of associated devices.

## AN EXAMPLE SCENARIO OF DISTRIBUTED USER INTERFACE

In order to explain how the environment supports the distribution specification, we consider an example application that can be dynamically distributed on a mobile and a public display in order to access games in a museum. We focus on the application part related to a game that shows pictures of some artworks in a museum and the user has to indicate which of them was not made by a given artist. The user interface is composed of a presentation that contains the application title, a grouping containing the navigator menu, a grouping containing all the artwork images suitably dimensioned for the device type, a text representing the question, a single choice with the possible answers and finally an activator to submit the answer to the server. The initial user interface (Figure 2) is rendered in a mobile device. Thus, the interface is associated with the assignment property, and the contained elements inherit the same CARE value.

Since mobile devices have small screens, at some point the user may want to distribute the user interface in order to show some information on a large screen, for example to share the main visual content with other users in order to decide together which reply to enter in the game. To obtain the user interface distribution shown in Figure 3 some CARE values have been changed through a Customization Tool (described afterward): the interface and the presentation are no longer assigned only to the mobile device, but are distributed in a complementary way on both devices, thus the CARE value associated with the interface and presentation is now complementarity.

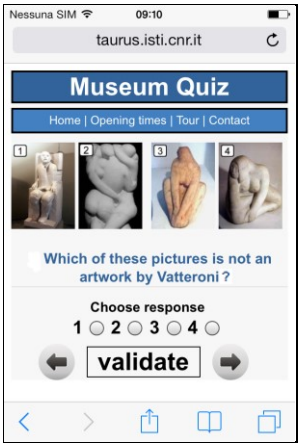


**Figure 2. UI completely assigned to a mobile device**

How the elements are distributed is specified in the child elements of the presentation by assigning the relevant CARE properties. In particular, the application title, the grouping containing the navigator menu and the question are redundant on both devices, while the grouping containing the images is assigned to the desktop device because it has a larger screen that can better show its content. The single choice is assigned to the mobile device and the activator element corresponding to the button with "Validate" label is decomposed according to its three subparts: input and prompt parts are assigned to the mobile device, while the feedback part is redundant on the two devices. Thus, the user can press the button on his own personal device and also show the results on the large screen shared with other users.

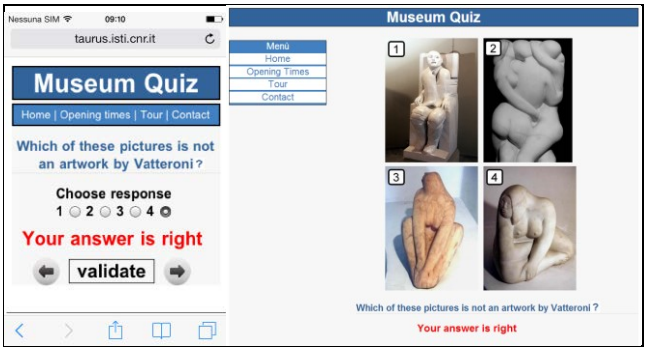


**Figure 3. UI distributed in a complementary way: part on mobile device (left) and part on desktop device (right)**

## THE LANGUAGE FOR SPECIFYING DISTRIBUTED USER INTERFACES

For the user interface distribution specification, we used as starting point the MARIA language [15], whose definition is publicly available. It has a modular approach with one language for the abstract description (AUI, Abstract User Interface) and then a number of platform-dependent concrete languages (CUI: Concrete User Interface) that refine the abstract one depending on the interaction resources of the considered platform. A platform is a type of device (e.g. desktop, mobile) that supports specific interaction modalities (e.g. graphical, vocal). Thus, for each element in the abstract language there is one or more refinements for the concrete platform considered. A user interface is composed of one or more presentations. Each of them contains a number of possible elementary interactors and compositions thereof. The language includes a data model, needed for representing the data handled by the user interface, which is shared between all the defined presentations. Currently MARIA provides concrete user interfaces for platforms such as: graphical desktop, graphical smartphone, vocal, multimodal desktop and multimodal smartphone. Thus, it opens up the possibility of obtaining distributed user interfaces that support various interaction modalities. However, prior to this work the MARIA language did not provide support for distributed user interfaces at all. Thus, we have introduced an original solution for modelling dynamic distribution integrated with MARIA, along with the associated tools for design and run-time support.

In order to introduce the distribution support we have considered the concrete level because determining any distribution requires considering the concrete features of the devices involved. For example, in some cases users may be interested in a distribution from a mobile to a stationary device in order to exploit a large screen, and aspects such as screen size and interaction modality are not captured at the abstract level, which is platform independent by definition.

The new concrete language for DUIs has to indicate how distribution occurs amongst devices that can belong to different platforms and how it can change over time. Regarding the description of the actual user interface elements in the various devices involved, it uses the existing concrete languages for the various platforms. We also added the possibility of describing the distribution at any granularity level (user interface, presentation, interactor compositions, single interactors, subparts of interactive interactors). Thus, when one such element is introduced it is possible to indicate the corresponding distribution properties (CARE value and target platform). We also added a new event type called Distribution Event, which can be used to specify at design time when dynamic changes in the distribution of the user interface elements should occur. The distribution event is associated with a handler in which the new values of the CARE properties are assigned, and it is possible to indicate new devices or platforms involved in the distribution.

Figure 4 shows an example regarding a city guide application. Such application allows the guides to distribute user interface components from their tablets to the smartphones of the tourists in such a way as to determine the most suitable interactive components also taking into account the visitors' abilities. Vision-impaired tourist use a multimodal version of the application for smartphones.

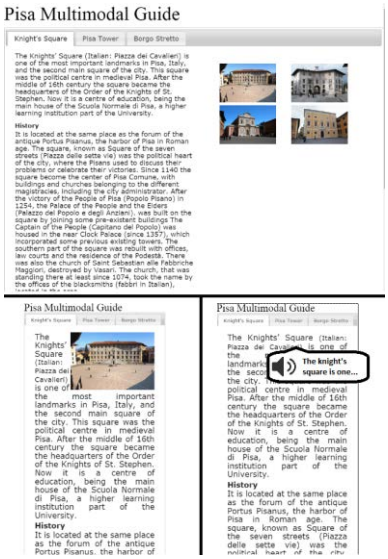


**Figure 4. Small example of distributed user interface.**

The excerpt in Figure 5 shows how it is possible to specify the distribution in the XML language: there is a grouping of elements contained in the current tab which is associated with the "complementarity" value since parts of them are for the mobile and tablet platforms and parts for another platform (multimodal mobile).

```
<tab_container>
    <tab_element>
    <label>Knights' Square</label>
    <content>
        <distribution care_value="COMPLEMENTARITY"/>
        <description>
            <distribution care_value="REDUNDANCY" >
                <device name="guide" platform="tablet"/>
                <device platform="mobile"/>
            <distribution>
            <description_desktop>
                <text>The Knights' Square (Italian: Piazza dei Cavalieri)...</text>
            </description_desktop>
            <description_mobile>
                <text>The Knights' Square (Italian: Piazza dei Cavalieri)...</text>
            </description_mobile>
        </description>
        <description>
            <distribution care_value="ASSIGNMENT" >
                <device name="visually_impaired" platform="multimodal_mobile"/>
            <distribution>
            <description_multimodal_mobile>
                <text>The Knights' Square (Italian: Piazza dei Cavalieri)...</text>
                <vocal_text>The Knights' Square, in Italian is called Piazza dei Cavalieri,
            </description_multimodal_mobile>
        </description>
        <grouping>
            <description>
                <description_desktop><image src="cavalieri.jpg"/></description_desktop>
                <description_mobile><image src="cavalieri_small.jpg"/></description_mobile>
                <distribution care_value="ASSIGNMENT" >
                    <device platform="mobile"/>
                <distribution>
            </description>
            <description>
                <description_desktop><image src="miniature_1.jpg"/></description_desktop>
                <description_mobile><image src="miniature_1.jpg"/></description_mobile>
                <distribution care_value="ASSIGNMENT" >
                    <device name="guide" platform="tablet"/>
                <distribution>
            </description>
            ...
```

**Figure 5. Excerpt of distributed user interface specification.**

The tab content contains a description element distributed in a redundant manner for both tablet and mobile platform because the textual description is shown in the same way

across them. There is one further description element assigned for the multimodal mobile platform that contains a textual and a vocal component in order to better support vision-impaired users. In addition, because of the user disability, this user interface does not contain the images that are shown in the other tourists devices.

The authoring environment (Figure 6) provides support for specifying the distribution. It allows developers to work by direct manipulation on a graphical representation in the main central area that is easier to read than the XML specification. In the right-hand panel (Distribution Tab) they can interactively specify the distribution element for any element at any granularity level by indicating the corresponding CARE value. In the left-hand part is also possible to interactively indicate the devices involved in the distribution. The generation software includes in the final implementation an instance of the data model, which will be used by the run-time support to store the state of the user interface and keep synchronised the user interface elements that are distributed on multiple devices. This allows, for example, that when a text edit element is redundant on two devices if the user changes a value in one then the other will show the updated value.

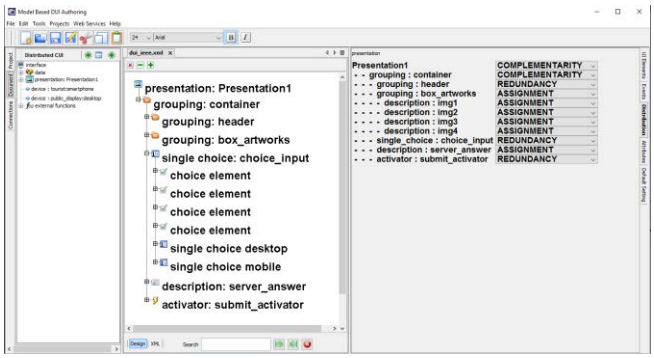


**Figure 6. The authoring environment for distributed UIs.**

## DISTRIBUTED USER INTERFACE GENERATION

We have designed and implemented the underlying support that transforms the XML description of the distributed user interface into a corresponding executable implementation. As previously mentioned, the distributed concrete user interface contains the description of each UI element for each platform involved in the distribution and the distribution profile. At the beginning of the generation process the original specification of the distributed user interface must be split into a number of Concrete User Interface descriptions for each target platform. During the splitting transformation the generator also derives a “Sync” module that contains the distribution profile (CARE values, devices and platforms involved in the distribution) for all user interface elements without their concrete descriptions for each platform. This splitting transformation is implemented through an XSLT style sheet. Then, the various parts of the XML-based specification are provided as input to the corresponding code generators for the target platforms, and there is also a generator for providing useful information for the software components in charge to handle the distribution at run-time. Each of the resulting platform-dependent concrete user interfaces is thus transformed into an FUI (Final User Interface) implemented in JSP (Figure 7). The generator also inserts in the final user interface some JavaScripts that will be used at run-time to dynamically receive new CARE values and change the visibility of the UI elements accordingly when the distribution state changes, and to keep the distributed user interface elements synchronized when a user changes an interactive element value in another device.

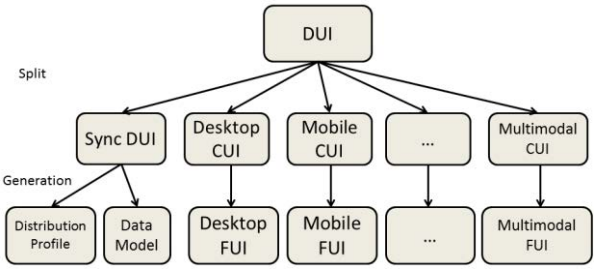


**Figure 7. Generation Process from the DUI specification**

All the user interface elements are generated, but when the resulting application is launched only those that should be perceivable according to the CARE properties defined at design time will be displayed. We took this decision because dynamic run-time generation and upload can impact the application performance and thus degrade the user-experience. To be more precise, the HTML code corresponding to all defined interface elements is generated, but their initial visibility depends on the initial value of the corresponding CARE property, which is used to set their CSS parameters that determine their display accordingly

Generation also involves producing code on the server side, which manages the Distribution Profile and the data model, and triggers the necessary updates on the clients. The generator takes the ”Sync” module containing only the distribution information and generates a CARE values table describing how each user interface element is currently distributed, thus implementing the initial distribution configuration. In the generation process the data model defined in the distributed CUI is also considered in order to obtain the corresponding run-time component (a Java class) for managing its values. The data model indicates data that the user can manipulate through the user interface, ranging from simple text field values to more structured elements such as calendars.

## DISTRIBUTION CUSTOMIZATION TOOLS

In general, in our approach it is possible to manage distribution in various ways: distribution specified in the model-based description of the interactive application with the initial specification of the CARE properties (design-time); distribution defined through the handlers of the distribution events indicated in the interactive application specification (design-time definition + run-time execution); distribution obtained through the dynamic customization

tool (completely run-time), which allows users to obtain distributions that were not planned at design time; distribution specified by direct UI manipulation (completely run-time), this possibility has been introduced after the user test reported in the following.
Regarding the possibility to manage the distribution at run-time, the solution offers a tool able to show the logical structure of the user interface and allow the users to dynamically change the distribution of its elements. For this purpose the users can control and modify, at any granularity level, the value of the CARE properties and the devices on which the elements considered are distributed (Figure 8).

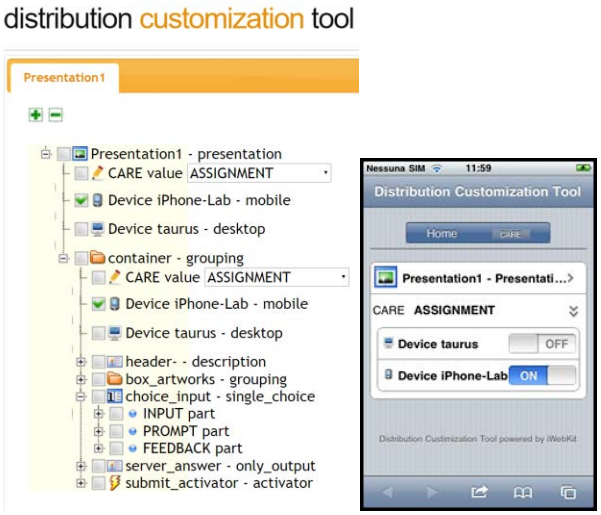


**Figure 8. Distribution customization tool for desktop (left) and mobile (right) devices.**

Such distribution customization tool exploits the underlying model-based language by showing the hierarchical structure of the interface, which is automatically obtained from the concrete specification. Figure 8 shows the two versions of the first distribution customization tool developed. On the left there is the desktop version that shows the entire hierarchical structure of the interface in a single presentation through an interactive tree-like representation in which the various branches can be interactively folded/unfolded. On the right, there is the version for touch-based smartphones, which has been designed taking into account the small screen available and presents only one level of the interface logical hierarchy at a given time.

The customization tools show the interface elements at the various granularity levels and, for each item, the corresponding current CARE value and the device(s) on which the item is distributed. A user can change the CARE values and select a new device to indicate how to change the distribution of the interface at runtime. Sometimes the interface elements that appear in the customization tool do not have intuitive names, since they are automatically generated from the concrete description, and it can be difficult to understand what user interface element they refer to, especially if they refer to elements in the user interface active on another device. Hence, we provide a feature through which if the user selects an element in the customization tool, then the corresponding element on the actual DUI is highlighted.

Such customization tools also allow users to pull some user interface parts on other devices to their own device: for this purpose it is sufficient to change the device assigned in the CARE property. A video showing a demo is available at http://www.youtube.com/watch?v=1NxcR-xaerk.

## RUN-TIME SOFTWARE ARCHITECTURE

There are three types of cases that the run-time architecture of the distributed user interfaces has to manage: initial user interface allocation according to the CARE properties specified in the application; synchronization of user interface elements duplicated across various devices when the user changes a value in one of them; dynamic changes in the distribution according to requests received through the customization tools or on application requests when some specific events occur.

In order to support the management of such cases there are also two server-side modules that communicate with each other: the **UI Manager,** which is a servlet that supports the synchronization of the user interface elements across the various devices by exploiting the data model (produced by the generator), which stores the state of the user interface elements; and the **Distribution Manager,** which is also implemented as a servlet, and is able to access the **distribution profile**. Such profile consists in the CARE Values & Devices table generated from the concrete distributed user interface description as described in previous sections, and the devices currently associated at run-time. Thus, for each user interface element it indicates the associated CARE value, the devices that can be associated with it, and the IP address of the current devices that can show it. The Distribution Manager can change the current CARE values according to either the values resulting from changes triggered by distribution events indicated in the specification, or the values received from the customization tools.

At run-time, at first the interface elements are distributed according to the CARE values and the devices described in the interactive application specification. In order to exploit the distribution environment, the devices have to register with it by providing their identifier and platform. If they are consistent with the devices indicated in the specification, their IP address is added to the distribution profile. When devices other than those planned for in the specification access the distribution environment initially they do not receive any part of the user interface and are added to a specific device list. In this case the customization tool is automatically updated and displays the devices in the list as possible target devices of the distribution. Only new devices belonging to a platform already considered in the concrete user interface description can be dynamically involved in the distribution, e.g. if a vocal platform is not

considered in the distributed specification, then the correspondent implementation version is not generated, and for this reason it is not possible to distribute the interface on such a device in the event that it dynamically accesses the distribution environment. In case a device with an exclusive assignment of a UI element leaves the environment or, more generally, a necessary device is not available then an error is generated. At that point, the end user can still change the relevant CARE properties in order to allocate the involved user interface elements to other available devices.

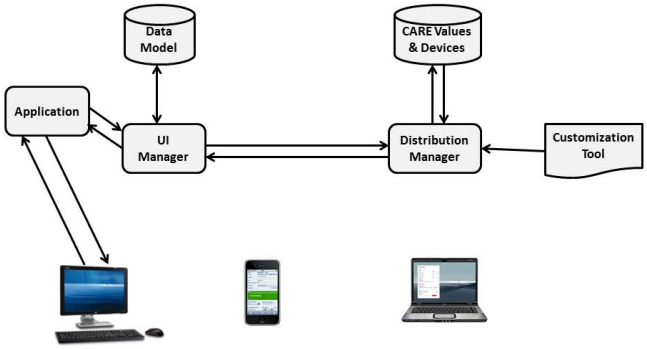


**Figure 9. Management of initial user interface configuration**

Whenever a client device accesses an application (Figure 9), the corresponding JSP is executed and first requests to the UI Manager the current values of the UI elements, then the current CARE values (and the associated devices) for each user interface element from the Distribution Manager, which is able to access the current distribution profile. Next, the application generates the user interface for the client with the elements that should be presented according to such profile. In case a composition of elements or some output only elements are not allocated to the device, then they are hidden. In case of interactive elements, if they are not allocated at all then they will be hidden completely. As mentioned before for such elements it is possible to have them partly shown on the user interface. Thus, there is a specific management process for such subparts. In particular, for the prompt subpart: if the value of the CARE property determines that the prompt is not distributed on the device, then the corresponding element is hidden through a JavaScript instruction, although it is still present in the page DOM; for the input subpart: if the value of the CARE property determines that the input is not distributed on the device, then the element is disabled through a JavaScript instruction; the feedback subpart is managed later on when an interaction occurs; if the feedback is distributed on the device considered, then that device will receive constant feedback updates otherwise it will not.

Synchronization of the state of the user interface elements duplicated across various devices is necessary when the user changes a value in one of them. When this happens the UI Manager is notified in order to first update the data model. Such model contains the data with the corresponding values (the state) of all interface elements, which are distributed on multiple devices. In order to make such synchronization possible some JavaScripts are included when the user interface is generated. In particular, each generated interactive element is associated to an event (on value change) and a handler, so that when a user changes the value of an element, such handler sends the new value to the server-side module (the UI manager), which updates the data model and propagates it to all involved users interface parts distributed in other devices. These user interface parts contain also scripts in charge of receiving the values modified in other devices and updating the local UI consequently. Thus, when a value is changed in the user interface this is communicated to the UI Manager (see 1 in Figure10), which is able to access the shared data model object and update it with it (2). This component also checks whether other devices are displaying the modified element through communication with the distribution manager (3), which is able to access the current distribution profile (4). In case in the distribution profile there is a CARE value of the feedback part of that element that is associated with other devices, this is communicated to the UI manager (5), which then sends this value to all involved devices (6). This is implemented through the Web socket mechanism[2], Figure 10 indicates the messages sent through the Web socket protocol with dashed lines.

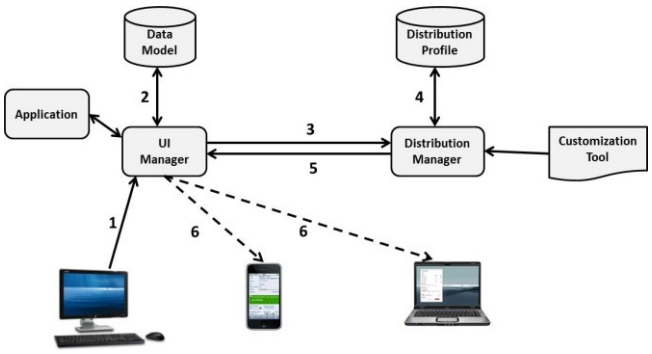


**Figure 10. Synchronization of cross-device UI state**

The software architecture also provides support when a user changes the distribution state through a Customization Tool. In this case, the new CARE values are sent to the Distribution Manager, which updates the current Distribution profile accordingly, and then sends the updates to the user interface components distributed across the devices. This can also imply showing previously hidden elements that become perceivable according to the new values of the CARE properties.

## EVALUATION

We have carried out a first user test to investigate the usability and the usefulness of the distribution platform, including the possibility of dynamically configuring the distribution. It was a formative evaluation, whose results have been useful to improve the environment, assess

[2] http://dev.w3.org/html5/websockets/

whether the CARE properties provide a useful vocabulary to control user interface distribution, and the usability and usefulness of the original customization tools in this distribution process.

The test involved 20 people, aged between 20 and 39 (M: 28.1, SD: 3.81), 13 of them were males and 7 females. 10 participants held a Bachelor Degree, 6 a Master Degree and 4 a High School diploma. Ten users work or study in the IT sector and ten have familiarity with Web browsing (they were more familiar with desktop devices than mobile devices) but they have no knowledge in software development. Only one user already knew another system to distribute (or migrate) a Web page to different devices, none of them had previously used our distribution platform. Before starting the test we gave a brief introduction of the relevant concepts and about the test aims. Users performed the test separately and one analyst took note of any errors or difficulties. During the test no help was provided. After the test, each user completed an evaluation questionnaire. The entire process took about 45 minutes per user.

In the proposed scenario the user accessed the application to play a game in a museum mobile guide. At the beginning users started the visit through their mobile device, on which the user interface was rendered. Users had to open the customization tool from the mobile device, change the presentation CARE property and duplicate the user interface in its entirety on a desktop device. Once the interface was replicated on both devices users could enter an input from the mobile device and see it on the desktop device as well. After that, users had to open the customization tool from the desktop device and distribute the presentation in a complementary way: all images had to be assigned only to the desktop device and the other interface elements had to be duplicated in both devices. Then users could interact with the application and see the feedback of their interactions on both devices. Finally, users had to interact with the customization tool from the mobile device to locate in the user interface representation the element corresponding to the single choice input element present in the user interface selected. Once the users found the indicated element they had to assign the input, prompt and feedback subparts to the mobile device.

All users were able to complete the tasks. In case of mistakes the moderator suggested that the users read the introduction text containing the definitions of the terms. For the evaluation questionnaire users had to rate various aspects of the distribution process on a scale from 1 to 5 (1 as the most negative score and 5 as the most positive one) and for each question they were asked to provide comments. Ratings are reported in terms of: range of ratings received, programming expert median (PE-M), not programming expert median (nPE-M) and interquartile range (IQR) that is calculated as the difference between the third and the first quartile (programming expert [PE-IQR] and not programming expert [nPE-IQR]).

*Intuitiveness of the customization tool [desktop version]* ([1-5], PE-M: 4; nPE-M: 4; PE-IQR: 4.75-4=0.75; nPE-IQR: 5-3=2)

*Intuitiveness of the customization tool [mobile version]* ([1-4], PE-M: 4; nPE-M: 3; PE-IQR: 4-3=1; nPE-IQR: 3.75-3=0.75)

Users found the desktop structure of the customization tool more intuitive than the mobile version; some of them reported some trouble using the mobile version because of the limited capacity to provide an overview of the interface due to the limited screen size.

*Ease of identifying UI elements within the customization tool [desktop version]* ([2-5], PE-M: 4, nPE-M: 4.5; PE-IQR: 5-4=1; nPE-IQR: 5-4=1)

*Ease of identifying UI elements within the customization tool [mobile version]* ([1-5], PE-M: 4, nPE-M: 3.5; PE-IQR: 4-3=1; nPE-IQR: 4.75-3=1.75)

Users found more difficult to find an element within the mobile version of the customization tool because this version shows only one level of the interface hierarchy at a given time. It may be better to have a greater similarity between the two versions in order to facilitate users.

*Usefulness of the highlight method in the customization tool* ([4-5], PE-M:5; nPE-M: 4.5; PE-IQR: 5-5=0; nPE-IQR: 5-4=1)

This aspect received quite good ratings: without the highlight method it would have been difficult to locate the UI element to distribute from the customization tool.

*Ease of performing the distribution process [desktop version]* ([2-5], PE-M: 4, nPE-M: 4; PE-IQR: 4.75-4=0.75; nPE-IQR: 5-4=1)

*Ease of performing the distribution process [mobile version]* ([1-5], PE-M: 4, nPE-M: 4; PE-IQR: 4.75-4=0.75; nPE-IQR: 4.75-3=1.75)

In general, participants thought that the distribution process is easy; some users reported troubles related to the difficulties finding an element within the mobile customization tool.

*Usefulness of CARE properties to describe the distribution* ([1-5], PE-M: 4.5; nPE-M: 4; PE-IQR: 5-4=1; nPE-IQR: 4.75-3.25=1.50)

Only one person gave the worst score (1 = Not helpful at all) commenting that the difference between the CARE properties values are too complex and it should be hidden to users; it would be better to choose other values like “show to all”, “show this element to this device”.

*Completeness of the CARE properties* ([3-5], PE-M:4; nPE-M: 4.5; PE-IQR: 5-4=1; nPE-IQR: 5-4=1 )

This section received quite good ratings; the only relevant comment concerned the Assignment value that could be also valid in case of multiple assignment, thus replacing the Redundancy.

*Clearness of the meaning of all CARE properties* ([1-5], PE-M:4.5; nPE-M: 4; PE-IQR: 5-4=1; nPE-IQR: 4-4=0 )

Three participants found the initial description too short and not very clear, but after using the application the meaning became clearer. Four users said that Equivalence is not the appropriate term and two of them initially thought that it had the same meaning as Redundancy.

*Clarity of the meaning of interactive subparts* (input, prompt, feedback) ([3-5]; PE-M: 4; nPE-M: 4.5; PE-IQR: 5-4=1; nPE-IQR: 5-4=1)

Some issues were raised about the prompt definition, which was not clear for some users; regarding the other two subparts there were no problems.

*Usefulness to decompose interactive elements in subparts* ([3-5] PE-M:5; nPE-M: 4; PE-IQR: 5-4=1; nPE-IQR: 5-4=1)

This section received quite good ratings; one user said that this granularity could be too verbose in some cases; it could be useful to allow users to specify a CARE property at the interactive element level, and thus each subpart would inherit it. We agreed that this was a good suggestion and have adopted it.

## DISCUSSION

The type of user test conducted was a formative evaluation, able to provide useful comments and feedback, rather than a summative evaluation. Overall, this evaluation provided positive feedback and suggestions for improvements, some of which have subsequently been implemented. The users indicated various applications in which distribution can be useful, almost all suggestions concerned collaborative applications. One participant proposed utilizing the tool to distribute only a portion of a Web site on a mobile device: it could be useful for Web sites that are not designed for mobile devices. The distribution is useful in contexts in which there are devices provided with large screens on which to distribute texts or images in order to exploit the large screen display. Two users observed that a useful scenario for the distribution could be an e-learning application where each student can access and see her own answers and the teacher can see all. One user suggested adopting the distribution platform for games involving multiple users with a super-user that can see the state related to all users and control the game.

In general, the evaluation of the customization tool received positive ratings; users gave a better rating to the desktop version; this could be explained by the fact that the desktop version displays the hierarchical structure of the entire interface, while on the contrary, the mobile version shows only one interface hierarchy level at a time. Regarding usefulness, completeness and clearness of CARE properties the user evaluations were positive and encouraging, people who do not work in the information technology field gave lower rating then the others who are more familiar with these concepts and thus able to better interpret them in this context. There were not substantial differences between the two categories of users considered. In general, programming experts ratings were slightly higher, hence those who work or study in the IT sector found it easier/more useful the distribution process than the others. However, the evaluation ratings were similar, showing that also people who are not computer expert can perform the distribution process through the proposed tool without particular problems. As a consequence of the evaluation we have introduced a further way to customize the distribution. The reason for this addition is that the user test highlighted some problems with the distribution customization tool, in particular the mobile one. Indeed, some users complained that they had to split the screen between the customization tool and the UI, in order to understand which tool element corresponded to which UI element. For this reason we decided to introduce the possibility that users distribute directly through the UI. Thus, in the generation phase from the model-based specification to the final implementation we have added the inclusion of some JavaScripts that allow users to interactively select the various components and visualize the possible CARE properties to associate to them. Then, the user can directly select the CARE value and device(s) on which to distribute the UI element. In addition, during the user test one user highlighted a problem regarding the number of devices on which to distribute the UI: if there are too many devices it could be difficult to distribute an element on all of them by selecting each one individually. For this reason, we have decided to allow users to distribute a UI element (or the whole UI) also by indicating a target platform (and not only specific devices). In this way the considered element will be distributed on all the devices of that platform type that are subscribed at the distribution environment at that time.

## CONCLUSIONS

We have presented a solution able to address a variety of distribution scenarios at various granularity levels through both push and pull mechanisms. One advantage of the model-based approach is that it is not constrained to one implementation environment. Thus, it opens up the possibility of more general and interoperable solutions, which is important for distributed user interfaces since they aim to exploit the wide variety of devices that can be encountered while users are on the move. One limitation of our approach is the effort required to learn the model-based language. However, the clear logical user interface structure obtained is also exploited at run-time to facilitate the possibility of dynamic end-user customization of distribution. These seem to be interesting results that justify the moderate effort necessary in the initial modelling part.

## REFERENCES


1. Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May, Richard M. Young. 1995. Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE Properties. In *Proceedings of INTERACT.* Lillehammer. 115-120.
2. Marco Blumendorf, Dirk Roscher, Sahin Albayrak. 2010. Dynamic User Interface Distribution for Flexible Multimodal Interaction. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction* (ICMI-MLML'10). Beijing, China, 20:1 - 20:8.
3. David Dearman, Jeffery Pierce. 2008. It's on my other Computer!: computing with multiple devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '08), Florence, Italy, 767-776.
4. Albertos M. Felix , Victor M. R. Penichet, José A. Gallud. 2011. Drag&Share: A Shared Workspace for Distributed Synchronous Collaboration in *Distributed User Interfaces for the Distributed Ecosystem*. Ed. London, UK: Springer, 125-132.
5. José M.C. Fonseca. 2010. W3C Model-Based UI XG Final Report. Retrieved January 26, 2016 from http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/
6. Luca Frosini, Fabio Paternò. 2014. User interface distribution in multi-device and multi-user environments with dynamically migrating engines. In *Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems* (EICS '14). Rome, Italy, 55-64.
7. Maria Husmann, Michael Nebeling, Stefano Pongelli, Moira C. Norrie. 2014. MultiMasher: Providing Architectural Support and Visual Tools for Multi-device Mashups. In *Proceeding of Web Information Systems Engineering* (WISE 2014). 199-214.
8. Brad Johanson, Shankar R. Ponnekanti, Caesar Sengupta, Armando Fox. 2001. Multibrowsing: Moving Web Content across Multiple Displays, in *Procedeengs of the 3rd international conference on Ubiquitous Computing* (UbiComp '01), London, UK, 346-353.
9. Andreas Lorenz, Clara F. De Castro, Enrico Rukzio. 2009. Using Handheld Devices for Mobile Interaction with Display in Home Environments. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI'09). ACM Press, NY, USA, 1-10.
10. Kris Luyten, Karin Coninx. 2005. Distributed User Interface Elements to support Smart Interaction Spaces. In *Proceedings of the Seventh IEEE International Symposium on Multimedia*. Washington, DC, USA, 277-286.
11. Ethan Marcotte. 2011. *Responsive Web Design, A Book Apart*. http://www.abookapart.com/products/responsive-web-design
12. Célia Martinie, David Navarre, Philippe Palanque. (2014). A multi-formalism approach for model-based dynamic distribution of user interfaces of critical interactive systems. *International Journal of Human-Computer Studies*. 72, 1. (January 2014), 77-99.
13. Jérémie Melchior, Jean Vanderdonckt, Peter Van Roy. 2011. A model-based approach for distributed user interfaces. In *Proceedings of the 3rd ACM SIGCHI symposium on Engineering Interactive Computing System* (EICS 2011). Pisa, Italy, 11-20.
14. Bread A. Myers. 2005. Using handhelds for wireless remote control of PCs and appliances. *Interacting with Computers*. 17, 3 (May 2005), 251-264.
15. Fabio Paternò, Carmen Santoro, Lucio D. Spano. 2009. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Transactions on Computer-Human Interaction* (TOCHI), 16,4 (November 2009), 1-30.
16. Antonio Penalver, José J. Lopez, José A. Gallud, Enrique Lazcorreta, Federico Botella. 2011. Distributed User Interfaces: Specification of Essential Properties. *Distributed User Interfaces - Designing Interfaces for the Distributed Ecosystem for the Distributed Ecosystem*. Ed. London, UK: Springer. 13-21.
17. Shankar R. Ponnekanti, Brad Johanson, Emre Kiciman, Aarmando Fox. 2003. Portability, Extensibility and Robustness in iROS. in *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications* (PERCOM '03). IEEE Computer Society, Washington, DC, USA, 11- 19.
18. Umar Rashid, Miguel A. Nacenta, Aaron Quigley. 2012. The Cost of Display Switching: A Comparison of Mobile, Large Display and Hybrid UI Configurations. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (AVI'12). Capri Island, Italy, 99-106.
19. Jishuo Yang, Daniel Wigdor. 2014. Panelrama: enabling easy specification of cross-device web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI 2014). Toronto, Canada, 2783-2792.