Authoring Context-dependent Cross-device User Interfaces based on Trigger/Action Rules

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

ABSTRACT
Current authoring environments provide the possibility of developing user interfaces with limited adaptation capacities. The most widely adopted tools follow the responsive design approach and allow developers to obtain user interfaces that can adapt mainly to the screen size and orientation. We present a solution able to support development of user interfaces able to adapt to the various types of contextual events (that can be related to users, devices, environments, and social relationships), with the added possibility of distributing the user interface elements across multiple devices. The context-dependent behavior is modelled through trigger / action rules, and can even be applied to Web applications that were not originally designed to be context-aware. This paper describes the design and main features of the novel authoring environment and reports on a first user study.

CCS Concepts
• Human-centered computing~Ubiquitous and mobile computing systems and tools; • Software and its engineering~Context specific languages; • Software and its engineering~Development frameworks and environments;

Author Keywords
Ubiquitous Computing; Context-Awareness; Cross-device User interfaces

INTRODUCTION
Ubiquitous computing is becoming reality, however developing applications that can actually exploit the rich technological offer in terms of devices and sensors and improve user experience is still difficult. Herein we focus on Web applications that can be accessed from any browser-enabled device, and currently the main approach for addressing the variety of possible devices is responsive design [11], which mainly consists of showing, hiding or changing user interface elements depending on the screen size of the available device or windows detected through media queries. However, this seems too limited since there can be various contextual changes that may require adapting the interactive application and, in some cases, it can be useful to distribute its user interface across different devices to facilitate transferring and sharing of information.

We consider the context of use structured along four main dimensions: the user (the tasks, the preferences, the emotional state, etc.), the devices (their interaction resources, connectivity, multimedia support, etc.), the environment (noise, light, temperature, etc.), and social relationships (friendships, groups, etc.). One of the main first attempts to provide support for the development of context-enabled applications was the context toolkit [17], which provided a library aiming to hide the complexity of the actual sensors. However, it considered a limited set of events and required a programming style that could be difficult to apply because it required developing code that is deeply intertwined with the application. We propose a more modular approach, with a clear separation of concerns, in which the role of application, context management and context-dependent adaptation are clearly distinguished, and their integration is precisely defined. Indeed, our approach is based on an authoring environment that allows developers and designers to interactively add adaptation rules modelled in terms of triggers and consequent actions, which can even be defined incrementally by people other than the original application developers in order to create different versions for context-dependent customizations. For example, it is possible to define versions that provide different customizations depending on the users' roles. In addition, with such context-dependent behavior it is also possible to make the user interface cross-device (with synchronized elements distributed across multiple devices) in such a way to exploit devices that are encountered while freely moving about, the typical example being when users find a public display and want to exploit it to share information from their personal device with others.

We envision various application domains that can benefit from such possibilities: for example, smart retail in which large shops can customize real-time support for the shoppers, city or museum guides in order to facilitate group visits with context-dependent information and games, learning applications with the possibility to adapt the contents and the way of presenting them depending on

MUM '15, November 30-December 02, 2015, Linz, Austria

DOI: http://dx.doi.org/10.1145/2836041.2836073

dynamic information on available devices and nearby people, and personal state.

In the paper, after discussion of related work we introduce example scenarios that can be addressed with our solution, followed by the main features of the authoring environment, we then illustrate the underlying software architecture and indicate how it is integrated with a context manager infrastructure and how the adaptation rules are applied to the corresponding applications. We also report on a first user test and discuss the positive aspects along with some suggestions for further improvements. Lastly, we draw some conclusions and provide indications about future work.

RELATED WORK

Our work draws from research on context-aware adaptation, multi-device design tools, and cross-device user interfaces.

Context-aware Adaptation

Providing context awareness to computer applications has been a challenge for decades. Stick-e notes [1] was one of the first attempts to make applications able to adapt to the context of use by specifying conditions. With the technology evolution, customizing user interaction in smartphones has quickly raised interest. For example, an early proposal [10] provided the possibility of defining context-action rules through which users can connect interaction inputs (contexts) to application actions in Symbian devices (e.g. when the user performs a circle gesture then the smartphone becomes silent). Various studies have shown that even the interaction modalities can change according to the context of use in order to better support users. Some of them are reported in [4], where they are classified depending on aspects related to environment (e.g. brightness, noise), social conditions (e.g. stress, social interaction, and location). The possibility of going beyond responsive design in order to consider various possible contextual events and then adapt the level of multimodality accordingly has been proposed in [6], in which rules expressed in terms of event / condition / actions were exploited for this purpose. A similar format has been exploited in Keep Doing it [13], a mobile application that continuously records users' interactions in such a way to allow users to automate a task based on their latest actions in a kind of programming by example approach. The contextual events that can be managed by this approach are those that can be detected through the sensors and peripherals of modern smartphones. In general, this type of approach has limited applicability, so it can be useful to automate short sequences of actions but cannot support more generally the development of context-aware applications. This type of issue has been addressed in [16] through an event-driven workflow framework to develop context-aware mobile applications. The types of events that it can detect are limited to locations, QR-codes, and time and they are used to trigger activities described in the workflow. Thus, overall it still does not support the authoring of various types of context-dependent applications. On the other hand, we can notice that there is a general trend to consider trigger / action programming to facilitate the development of applications reactive to contextual events. Indeed, there is the IFTTT environment[1] that facilitates the creation of recipes that indicate actions to perform when some change occurs in frequently used social network applications. IFTTT only supports recipes composed of one event and one action. A recent study [18] has also found that users found an extension of such language easy to use to model small contextual home applications even by people with limited programming experience.

Multi-device Design Tools

One of the first tools addressing authoring of multi-device user interfaces was Damask [9]. It used the concept of layers to indicate parts of the user interface that can be associated to either one specific device type or to all device types, and exploited a set of patterns with the possibility of sketching the desired user interface in order to facilitate its development. Another tool in this area was Jelly [14] that did not use layers but still enabled designers to copy components across devices, and when an element was copied designers could select from a list of available widgets how it should look on the other device. Another difference was that Jelly focused on creating running user interfaces on top of existing toolkits instead of sketching low fidelity prototypes.

In the meantime, with the advent of responsive design various tools for creating applications according to this approach have been put forward. An example is Webflow[2] that facilitates the specification of different stylesheets depending on the media queries and provides a number of responsive website templates. In general, these approaches have mainly considered multi-device applications in which the user actually exploits only one device at a given time to access the application. An attempt to address even the authoring of distributed user interfaces in which at a given time the user interface is distributed across multiple devices is XDStudio [15]. It supports two complementary authoring modes: simulated and on-device. In the former mode, authoring is carried out on a single device in which the user interfaces distributed on other devices are simulated. In the latter mode, design and development actually takes place on the target devices themselves. However, this type of authoring environment does not provide support for specifying context-dependent behavior, which is an important feature supported by our environment.

Cross-device User Interfaces

In recent years some frameworks that provide useful support for developing cross-device user interfaces have been proposed. The proximity toolkit [12] simplifies the

[1] https://ifttt.com/

[2] https://webflow.com/

exploration of interaction techniques by supplying fine-grained proxemics information between people, portable devices, large interactive surfaces, and other non-digital objects in a room-sized environment. It facilitates rapid prototyping of proxemic-aware systems by supplying developers with the orientation, distance, motion, identity, and location information between entities, including a visual monitoring tool that allows developers to visually observe, record and explore proxemic relationships in 3D space. Its architecture separates sensing hardware from the proxemic data model derived from these sensors, which means that a variety of sensing technologies can be substituted or combined to derive proxemic information. We adopt a similar separation in order to gather contextual information from a variety of sensors.

Specific aspects related to how to minimize seams in interaction with multiple devices by dynamic alignment between interfaces have been addressed in [7].

A framework supporting user interface distribution in multi-device and multi-user environments with dynamically migrating engines has been proposed [5]. It does not require a fixed server to manage the distribution. The elements of the UI can be distributed by specifying specific device(s), group(s) of devices, specific user(s), and groups of users according to roles. Panelrama [19] is a solution able to categorize device characteristics and dynamically change UI allocation to best-fit devices. For this purpose, this framework lets developers to specify the suitability of panels to different types of devices. This allows its optimization algorithm to distribute panels to devices that maximize their match for the developer's intent; as devices are added or disconnected, panels are automatically reallocated according to its optimization scheme.

The increasing availability of wearable devices in the context of cross-device user interfaces has been addressed by Weave [3], a framework for developers to create cross-device wearable interaction by scripting. It provides a set of JavaScript- based APIs to easily distribute UI output and combine sensing events and user input across mobile and wearable devices. It also has an integrated authoring environment to program and test cross-device behaviors and, when ready, deploy such behaviors. Similar frameworks aiming to provide structured support when developing applications involving smartwatches have been proposed in [2] and [8].

Our authoring environment draws inspiration from all these works, but extends existing concepts for context-dependent cross-device user interfaces through contextual trigger / action rules that can be edited by direct manipulation even on existing Web applications, and can also be exploited to obtain dynamic user interface distribution across multiple devices. Thus, it covers various aspects in an integrated approach that facilitates development and customizations of the target applications, and can be deployed in various settings.

SCENARIOS

In this section we describe two possible scenarios supported by our solution. In both scenarios, run-time context-awareness is addressed by a rule-based approach at authoring time. However, they are different since in the first scenario a single mobile device with context-dependent behavior is involved at run-time, while the second is characterized by cross-device interactions triggered by contextual events or on user request.

Walking Shopping List

A large supermarket provides its customers with a mobile shopping list application. Users can install the app in their smartphone and define the shopping list by selecting items available in the store before leaving home. When walking through the store in search of such items, the app provides various information on the items, such as position (e.g., the shelf number), price, ingredients, alternative and complementary products.

The marketing manager of the store is in charge of improving user experience and increasing sales. To this aim, s/he relies on a developer using the authoring tool for adaptation rules that allow them to define how the shopping list application will adapt according to contextual factors. One rule takes into account the customers' physical activity (detected by the device accelerometers) and shows additional information about the desired items (e.g. allergens, suggested recipes) or alternatives to them when the user walks slowly (indicating that they have time and interest to get additional information). When the walking speed increases, indicating that the user is in a hurry, the rule hides any additional content and emphasizes the most relevant information: the exact location of the currently selected item is displayed and the item picture is enlarged in order to facilitate the search in the shelf.

The application can also take into account additional contextual aspects, such as the proximity of an area (detected by monitoring the Bluetooth beacons nearby), in order to display advertisements "tailored" to the user profile. For example, personalized graphical/vocal advertisements about an aftershave, a shampoo or a perfume (depending on customer's gender, age) on discount are triggered when the customer walks slowly along the cosmetics aisle.

Tourist City Guide

A tourist guide regularly brings groups of people across an historical town and relies on an interactive application that acts as a multimedia support. The application contains information about aspects of interest related to the town (events, dates, famous people, pictures and videos, etc.). When organizing a tour, it is possible to create a set of custom adaptation rules taking into account the type of audience (adults, children, students) and their interests in order to define how to adapt the application to better exploit public displays deployed in the main points of interest such as the town hall, the archeological museum and the modern art gallery, and to show customized content to the tourist

version of the mobile guide. For each point of interest with a public display, the designer creates a rule that will trigger the distribution of parts of the application from the mobile device to the public display, in order to provide the audience with additional multimedia resources. The rule trigger is the vicinity of the public display. For instance, resources about the history of the municipality will be shown in the public display of the city hall as soon as the user mobile device detects the Bluetooth of the public display. Different sets of rules, with the same trigger but differing in the actions, can be defined for different classes of visitors. For instance, while texts and images could be distributed in case of adult audience, entertaining videos will be distributed instead if the audience is made up of schoolchildren. In addition, the guide version of the application can push some specific content to the tourists, if they so wish.

AUTHORING TOOL

The authoring tool was specifically designed for supporting the development of context-dependent cross-device user interfaces by defining rules for the application adaptation and distribution. The authoring environment is based on three main features: first, there is a clear distinction between the part dedicated to the user interface composition and that for the specification of the contextual rules. Second, the rules are structured in terms of triggers and associated actions, with the possible events and conditions defining the triggers classified according to four dimensions (user, device, environment, social), and the actions indicating how the user interface should change for the platform considered (so far we consider smartwatch, smartphone, tablet, PC, wide screen). Third, dynamic distribution of user interface across various devices can be indicated. Such distribution can be triggered by contextual events (e.g. when the user is close to a public display then some parts of the user interface are shown on it as well) or on explicit user request (UI events).

Tool Walkthrough

Figure 1 shows the overall authoring environment in two typical use cases. The main central area is where the user interface is composed for the currently selected platform. It shows the platform screen with inside the application user interface, which is adapted accordingly because the application version loaded is the one related to the chosen platform. Currently, five platforms (desktop, smartphone, tablet, smartwatch, and public display) are supported and those relevant can be selected in the graphical vertical menu on the left. In the application under development some scripts are included in order to facilitate the selection of the user interface parts to be adapted by direct manipulation.

On the right side there is the part of the authoring tool dedicated to the editing of the trigger / actions rules. The trigger / action rules approach is consistent to the event-condition-action (ECA) paradigm. There are two main types of events: the standard events that can be generated by a Web user interface (click, focus, mouse enter, change, etc.) and the contextual events, which are those mainly considered in this paper. As we mentioned, the aspects related to such contextual events are grouped along four dimensions: users (knowledge, task, disability, position, personal data, physical activity, proximity, etc.), environment (light, noise, temperature, structure, etc.), technology (devices, screen sizes, battery, connectivity, relative position, etc.), social (group memberships, level of friendships). Thus, developers can freely choose some contextual event and then indicate the possible effects. The top part of Figure 2 shows more in detail the selectable users dimension aspects. The elements with folder-shaped icon are entities (e.g. “disability”) and contain attributes (e.g. “blindness”) which have a sheet-shaped icon.

For specifying the actions the users can interactively select a part or an element of the user interface and indicate on which device types it should be visible or not or how some user interface attributes (such as colours, fonts, etc.) should change. Alternatively, a possible action can be the loading of a new page or the change of the content shown in the user interface part selected.

The rules edited can be saved and associated with the application, so that the developer can at any time preview the effect of their performance. For this purpose on the top part of the environment there is a list of rule triggers currently defined for the application under development, and by selecting one of them it is possible to simulate the contextual event and get a preview of the effect on the user interface. If the action of a rule specifies a distribution, then the main area is divided by the number of device types involved in order to show how the user interface is distributed across them. By selecting the triggers in the top part it is possible to see the effects in any of them. In this case, on the bottom side the authoring tool also shows the distribution profile, which consists in the indication of the device types involved.

The upper side of Figure 1 shows an example of adaptation rule definition for a smart shopping application. The user has selected the upper container (identified as “shoppingListContent” under the Actions part) and has set “font-size:25px” in the Update UI field. At run time the rule will increase the font size of the texts in the shoppingListContent element.

An example UI distribution definition for a tourist guide application is shown in the bottom side of Figure 1. The main part of the authoring tool displays the preview of a previously defined distribution rule, triggered by selecting the button in the top-left part of the interface (“Point_of_interest = Piazza della Signoria”). The distribution takes place when the vicinity to the point of interest is detected, and consists in some content (a textual description of the square) being distributed from the tablet device of the tourist guide to the smartphones of the group of tourists.

Figure 1. The authoring environment for context-dependent user interfaces: adaptation rule (top) and distribution rule (bottom) editing.

Implementation

The authoring environment is Web-based. On the main screen, the user can load an existing Web site via local or remote URL, which will be used as the source interface to define the context-dependent adaptations and distributions. We also defined a Chrome Extension (similar extensions can be implemented also for other browsers), which allows the tool to load an application user interface inside an Iframe in the Authoring Environment. The browser extension changes the User Agent of the Iframe depending on the currently selected platform. It is thus possible to present the different (and adapted) versions of the user interface according to the virtual device in use. Selection of the user interface elements to be adapted by a rule is managed by a script injected in the Iframe by the browser extension. This strategy avoids possible problems due to violations of the same origin policy, i.e. it allows the environment to interact with the Iframe content/functions also when it has a different domain from the authoring tool (e.g. when the application loaded in the Iframe is hosted in a different server).

When an element is hovered by the mouse pointer, the injected script sets its background to red and, if the element is selected, sets its border to red (see for example Figure 1, top-left, in which an item of the shopping list has been selected). The identifier of the selected element is shown in the "What" field of the "Actions" part (see Figure 2). The element selected is the one that will be affected by the updates specified in the "Actions" part.

The developer defines the adaptation/distribution trigger by firstly selecting an attribute from the contextual aspects tree. Such a structure is dynamically generated by the authoring tool according to the context schema retrieved in real time from the context model manager. The context schema is an XML Schema Definition (XSD) file describing the contextual resources in terms of the data type of the attributes contained in the various entities involved and in terms of the connections between the entities. The tree is

dynamically generated every time the authoring tool is opened. This allows, in case of modifications of the context schema, to have the tree in the authoring tool consistent with the context model manager automatically. Modifications in the context schema can be due to upgrades devoted to manage novel sensors embedded in newer smartphones (e.g., temperature, altitude, etc.) and/or additional user profile attributes, for example relevant for marketing aims.

Figure 2. Detail of the part for editing triggers and actions.

RUN-TIME ARCHITECTURE

In order to correctly execute the applications according to the adaptation rules specified it is necessary to have a specific support at run-time. The main goals of such support are to manage and apply the adaptation or distribution rules, and detect the events that trigger their performance. Such run-time support exploits the functionalities of three components:

- The context model manager is composed of a context server and a set of external modules delegated to monitor relevant parameters of the context of use (e.g. environmental noise, device coordinates, user physical activity). The purpose of the context model manager is to detect contextual events and inform those modules that subscribed to them. The context model manager shares the context schema with the authoring tool. This enables the authoring tool to display (see the upper part of Figure 2) exactly the contextual aspects that can be tackled at run time, so that the developer can define effective triggers;
- The distribution manager, which manages user interfaces distributed across multiple devices in order to allow dynamic migration of components and keep their state synchronized;
- The adaptation engine, which stores and manages the adaptation rules.

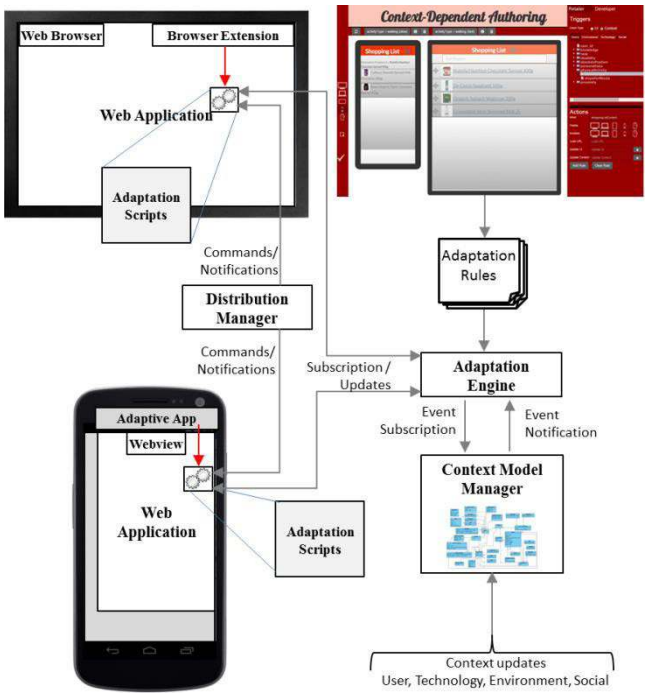

Figure 3. The architecture of the run-time support.

Figure 3 shows how such components interact with each other. The adaptation engine subscribes to the context model manager in order to be informed of the occurrence of the events relevant for the rules associated with the active applications. When one or more of such events occur, the adaptation engine sends the actions to the Web applications in order to perform the corresponding adaptation. Such updates commands are JSON encoded and are interpreted by the scripts included in the Web application by the authoring environment. They can modify properties of user interface elements or content, activate functions or navigation, etc. Some of such actions can even change the distribution of some user interface parts across devices, in this case the script in the Web application sends a corresponding command to the distribution manager, which notifies the involved devices. Such distribution manager contains the current distribution profile, which indicates how the various parts of the user interface are currently distributed across the devices that have subscribed to the environment. A distribution command mainly indicates that a user interface element or the elements included in a container should be visible or not on one specific device or a group of devices that have the same role or on all devices of a given platform.

DOMAIN-DEPENDENT EXTENSION

In order to facilitate the adoption of our authoring environment even by people who are not particularly expert in programming, we have also created an additional layer that provides support for creating rules that are particularly relevant in specific domains.

The basic idea is that the structure of a set of rules that can be frequently used in the considered domain is already defined and the application designer has just to specify the values for the specific case under consideration.

We have created an example of this domain-dependent extension for the smart retail area. The idea is to facilitate the creation of applications that can be exploited by shoppers while freely moving directly by the manager or the marketing expert of a large shop.

Figure 4. The domain-dependent support for the smart retail.

Figure 4 shows on the right a set of predefined rule structures that can be selected: "when the user is near …", "when the user is moving …", "when the user is entering-exiting …", "when the weather is …". Once the designer selects one of them then the specific parameters to define are graphically represented in the main central area. For example, if the rule selected was "when the user is near …" then the choice between a point of interest or a product or a store aisle is proposed, and after selection of one of them the available options in the current applications are indicated for completing the definition of the trigger. Then, the possible meaningful actions for the considered rule are shown. In the example they can be showing a video or a promotion or a message, and again the user can then complete the rule composition by selecting the relevant values.

USER TEST

The user test aimed to assess usability, usefulness and completeness of the environment. It did not consider the domain-dependent part, and thus it involved people with medium-high Web programming abilities.

Set up

Before interacting with the authoring tool, the participants could read an introduction about it, describing both the aims and the way the tool works. Then they watched a three minute video showing some examples of how the authoring tool can be used. After that, they were allowed to freely interact with the authoring tool for creating some rules (without any constraint on the triggers nor on the actions). Finally, they were asked to carry out the tasks related to two scenarios, one implying UI adaptation and one implying UI distribution.

The adaptation scenario was about an interactive shopping list application that had to be made adaptive according to the customer's physical behaviour. The users created two adaptation rules taking into account the customer's walking speed. The first rule, triggered when the customer walks fast, hides the additional products information and increases the font size of the main product information. The second rule is triggered when the customer walking speed is low. It restores the original layout and content, i.e. shows the additional information section and decreases the font size of the main information part.

The distribution scenario regarded the e-learning domain and was carried out on an online course hosted by Moodle[3] (which is the most popular Learning Management System). The main content of the course had to be made distributable based on two distribution rules taking into account the teacher position. In the first rule, one relevant part disappears from the teacher's smartphone and appears on the large screen of the classroom when the large screen is in proximity. The second rule restores the initial configuration, i.e. hides the distributed part on the large screen and makes it visible again on the smartphone when the system detects that the teacher has entered the teachers room.

The total test duration (reading instructions, watching video, familiarizing with the authoring tool and performing the requested tasks) was recorded for each participant, as well as the time taken for carrying out each one of the two scenarios.

After the interaction, the participants were requested to fill in an online questionnaire providing personal data including education and technical background, and a feedback on the tool. Quantitative ratings were given to assess the tool usability, usefulness and completeness, while some open-questions allowed to provide more general considerations and recommendations.

Participants

Twelve individuals were involved in the test, 5 female and 7 male with age between 26 and 45 (mean: 32.3, std. dev.: 5.12). One of them held a PhD, 4 a Master Degree, 6 a Bachelor and one a High School diploma. They were recruited in our Institute but were not involved in the design and development of the authoring tool, and the test was for them the first chance to try it. They rated their skills in Web programming on a 1 to 5 scale (5: excellent; 4: good; 3: average; 2: low; 1: none), between 2 and 5 (mean: 3.5, std.

[3] https://moodle.org/

dev.: 1.0). Half of the participants performed first scenario A and then scenario B, while for the others the order was inverted. This was done in order to reduce possible biases due to the learning effect when analysing users performance on the two scenarios (i.e. adaptation vs. distribution).

Three users had previously used an authoring tool and, among them, only one had used an environment for allowing UI distribution over multiple devices based on the context of use (Atooma for Android).

Results

We logged the total test duration for each user as well as the time taken for performing the two scenarios. All values are expressed in minutes. The total duration (including reading the instructions, watching to the video tutorials, familiarizing with the authoring tool and performing the two scenarios) varied between 26 and 49 minutes (mean: 37, std. dev.:7). The time to complete scenario A was between 4 and 15 (mean: 9, std. dev.: 3), while for scenario B it varied between 2 and 5 (mean: 4, std. dev.: 1). On average, the time spent to perform the distribution scenario was less than half of the time taken by the adaptation one. We did not run tests for proving statistical difference in the times, which would have been questionable due to the small sample size. However, we can motivate such a difference by observing that users had to explicitly write down the actions in the adaptation scenario (and this implied to focus on the proper CSS syntax). In the distribution scenario, they had simply to select some elements and then press some buttons to define elements (in)visibility in the various devices.

We asked users to rate, on a 1 to 7 Likert scale (with 7 as best score), the following aspects characterizing the proposed approach and the associated tool:

- Usability of the mechanism for selecting the rule trigger; min: 3, max: 7, mean: 5.3, med.: 6, std. dev.: 1.2;
- Usability of the system for defining rule actions; min: 2, max: 6, mean: 4.8, med.: 5, std. dev.: 1.2;
- Usability of the rule-based approach, in general; min: 4, max: 7, mean: 5.8, med.: 6, std. dev.: 1.0;
- Completeness of the set of events and actions that can be chosen; min: 3, max: 7, mean: 5.6, med.: 6, std. dev.: 1.0;
- Usefulness of the proposed approach for enhancing applications with context-awareness; min: 4, max: 7, mean: 5.8, med.: 6, std. dev.: 1.1;
- Usefulness of the proposed approach for making applications cross-device; min: 4, max: 7, mean, 5.3, med.: 5, std. dev.: 0.9.

Thus, overall the ratings were positive. The most appreciated aspect was the usefulness for obtaining context-aware applications, the lowest ratings were given to the usability in specifying the actions associated with the rules.

The participants could also provide observations and recommendations by answering to the following open questions.

What would you suggest to improve the usability of the proposed approach?

Three users noticed the lack of a clear feedback during rule creation, and recommended to show the updated list of actions attached to the rules as soon as they are specified. Another issue was due to the lack of a support for editing previously defined rules.

One user would simplify the entire interface because she considered it to be too cumbersome, for instance by allowing the selection and binding of multiple elements to one action. Another user would make the contextual entity names displayed on the tree structure more intuitive.

Would you add or remove any element from the set of events and actions?

One user declared she would add contextual information about the gyroscope to the context model.

Regarding the event definition, one user would like the list of operators for defining event constraints to be filtered according to the semantic of the aspect involved in the condition. For instance, the operators “lower than” or “greater than” may not be used for a condition on the identifier of a Point of Interest, and the operator “equal to” should be used instead.

Please cite example applications for which this approach can be particularly useful.

The participants mentioned applications that optimize online published content (e.g. books, newspapers) for the device in use, city/museum guides, supports for meeting presentations, systems for e-learning and professional training, domotics, healthcare (e.g. services for the elderly), online shopping and smart retail were among the various examples provided.

We asked the participants to mention three positive and three negative aspects of the authoring tool, including recommendations for general improvements.

Among the positive aspects, seven users mentioned the ease of use, five the adaptation/distribution preview capabilities, three the ease of device-oriented selection for specifying UI elements (in)visibility, and three the flexibility of the rule-based approach and the large field of application.

Most of the negative aspects were due to small lacks in the user interface layout or in the set of functionalities of the Authoring Tool. For instance, few users did not find intuitive the operators of the conditions because abbreviations were used, e.g. gt, lt, eq, etc. The absence of tips for specifying the UI updates based on CSS property changes was an issue for some users that would like to see a list of possible properties. Some users complaint about the

lack of continuous feedback during the rule creation phase (e.g. chosen trigger, defined actions). One user mentioned the impossibility of seeing the value of the property in the current interface while specifying the action to modify it (to this aim he relied on the browser debugger). The need for defining the same action for several elements instead of applying the same action once to a multiple selection of elements was also seen as problematic.

Besides the indications for improvements in the UI layout of the authoring tool, we collected an observation from one user regarding the UI state during multiple adaptations, i.e. sequential trigger of several rules. The user proposal was to have an automatic restoration of the original UI state just before triggering a rule. The aim would be to apply the actions of the rule to the original version of the UI, rather than on the current state (that may result from actions of previously triggered rules).

The following were among the most positive and encouraging comments: "It looks like a very good approach for managing context-awareness as it is intuitive and easy to use.", "The tool seems to be effective and quite easy to use.", "It is easy to use and lets you see the effects of your choices immediately in order to modify them if something wrong was done.", "It is intuitive and has high potentials for speeding up programming.", "It is easy to learn the mechanisms and the UI is intuitive."

We have saved the adaptation and distribution rules created during the test in order to subsequently analyse them. Regarding the trigger, the users could freely choose a contextual attribute and set a condition on it for triggering the rule.

We looked at the users' choices in order to quantify how many of them had actually created semantically valid conditions for the trigger.

In the adaptation scenario, seven users relied on the "steps per minute" attribute for expressing the walking speed, indicating a numerical threshold (e.g. greater or lower than 100). Four users chose the "activity type" attribute and picked "walking (slow)" or "walking (fast)" predefined values. One user used both attributes. All the users were thus able to create meaningful triggers in the adaptation scenario.

In the distribution scenario, the users were supposed to consider proximity of a large screen device in the first rule, and proximity of a point of interest in the second. Six users chose the right attributes for both rules. One specified the first rule correctly but selected the "task name" attribute for the second, and three selected "task name" for both rules (e.g. "task name = lecture", "task name = breakout"), which would be a different way to model the expected behaviour. Two users made invalid rules for detecting the entrance in the teachers room, considering proximity detection of a user or a device instead of a point of interest (the teachers room). Such mistakes were probably due to low confidence with the context model schema, and we believe that some short annotation of the context entities and attributes can better support novices in choosing the proper context aspect for the rule trigger.

During the test, we observed the participants interacting with the authoring tool and took note of the major issues they experienced. The mistakes that often led to malfunctioning rules confirmed the difficulties that some participants mentioned in the open questions of the questionnaire. For instance, at the first attempt, some participants created rules that did not apply the desired updates to the UI as expected or that did not work at all due to one or more missing actions. The reason was that they forgot to add the action to the rule and saved the rule with a trigger but without actions, or used a wrong syntax in the action (e.g., "font-size=10px" instead of "font-size:10px").

Most errors occurred during the initial familiarization phase the users had with the system, just before starting the real test session. However, by considering these problems and users' recommendations, we assume to be able to make the authoring tool easier to use also for novices and more robust with little effort. To this end we will enhance the system feedback at rule creation time, and add a syntax checker for the actions.

DISCUSSION

By looking at the results of the user study reported above, we are quite optimistic for future releases of our authoring environment. Although several participants complaint about missing functionalities and recommended some improvements, it appears that all of them were able to understand the main points of the approach. They indeed understood the semantic of the adaptation/distribution rules and were finally able to carry out the steps for their creation, namely trigger and actions definition. It is worth considering that 9 out of 12 participants declared not to have previously used any authoring environment, even if all of them had some skills on Web programming.

Other aspects that is worth to mention are expressivity and simplicity of use of the tool. Regarding expressivity, we assume that the authoring tool allows developers and designers to manage a good range of modifications to the user interface. Through the tool it is possible to define actions that change the appearance of any element or its contents or the navigation to different pages. The underlying language for the adaptation rules allows them to declare actions for any manipulation of the user interface (creation/update/deletion of elements, also with the support of conditionals and loops). We have however kept this first version of the authoring tool simple to use avoiding the possibility of creating particularly complex adaptation rules. Given the user test results, we believe the tool has a good tradeoff between expressivity and ease of use.

CONCLUSIONS

We have presented an authoring tool for supporting the development of context-dependent user interfaces, which is able to adapt and distribute themselves across multiple devices based on contextual events.

The user study we have carried out to evaluate the first version of the tool has shown the benefits of the trigger / action paradigm for defining the context-dependent adaptation and distribution rules. Participants found this solution simple and quick, and the proposed approach, in general, useful to address emerging scenarios characterized by contexts of use with a wide availability of devices and sensors.

We will dedicate future work to improving the authoring tool based on users' recommendations and adding further features. We will start by improving usability of editing the action part of the rules, e.g. by allowing multiple selection of elements, adding a suggested list for the CSS properties and syntax check for the updates. We will provide more support to define rule templates for the domain expert level, and carry out user tests for this part as well.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the Street Smart Retail project (grant n. 14607, European Institute of Innovation and Technology) for partially supporting this work.

REFERENCES

1. Brown, P.J. 1996. The Stick-e Document: a Framework for Creating Context-aware Applications. ElectronicPublishing, WILEY, Chichester, GB, vol. 8, no. 2-3, 24 September 1996, pp. 259-272.
2. Chen, X.A., Grossman, T., Wigdor, D.J., and Fitzmaurice, G. 2014. Duet: exploring joint interactions on a smart phone and a smart watch. Proceedings of CHI 2014, ACM, pp. 159-168.
3. Chi, P. and Li, Y. 2015. Weave: Scripting Cross-Device Wearable Interaction. Proceedings of CHI 2015, ACM, pp.3923-3932.
4. Dumas, B., Solórzano, M., and Signer, B. 2013. Design guidelines for adaptive multimodal mobile input solutions. Proceedings of MobileHCI 2013, ACM, pp. 285-294.
5. Frosini, L. and Paternò, F. 2014. User Interface Distribution in Multi-Device and Multi-User Environments with Dynamically Migrating Engines. Proceedings of EICS 2014, ACM, pp. 55-64.
6. Ghiani, G., Manca, M., Paternò, F., and Porta, C. 2014. Beyond Responsive Design: Context-Dependent Multimodal Augmentation of Web Applications. Proceedings of MobiWIS 2014, LNCS Volume 8640, pp. 71-85, Springer Verlag.
7. Grubert, J., Heinisch, M., Quigley, A.J., and Schmalstieg, D. 2015. MultiFi: Multi Fidelity Interaction with Displays On and Around the Body. Proceedings of CHI 2015, ACM, pp. 3933-3942.
8. Houben, S., and Marquardt, N. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. Proceedings of CHI 2015, ACM, pp. 1247-1256.
9. Lin, J. and Landay J. A. 2008. Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In Proceedings of CHI 2008, ACM, pp. 1313-1322.
10. Korpipää, P., Malm, E., Rantakokko, T., Kyllönen, V., Kela, J., Mäntyjärvi, J., Häkkilä, J., and Känsälä, I. 2006. Customizing User Interaction in Smart Phones. IEEE Pervasive Computing 5(3): 82-90 (2006).
11. Marcotte, E. 2011. Responsive Web Design, A Book Apart (2011), http://www.abookapart.com/products/responsive-web-design
12. Marquardt, N., Diaz-Marino, R., Boring, S., and Greenberg, S. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In Proceedings of UIST 2011, ACM, pp. 315-326.
13. Maues, R.A. and Barbosa, S.D.J. 2013. Keep doing what I just did: automating smartphones by demonstration. In Proceedings of MobileHCI 2013, ACM, pp. 295–303.
14. Meskens, J., Luyten, K., and Coninx, K. 2010. Jelly: a multi-device design environment for managing consistency across devices. In Proceedings of AVI 2010, ACM, pp. 289-296.
15. Nebeling, M., Mintsi, T., Husmann, M., Norrie, M. C. 2014. Interactive development of cross-device user interfaces. In Proceedings of CHI 2014, ACM, pp. 2793-2802.
16. Realinho, V., Romão, T., and Dias, A. E. 2012. An event-driven workflow framework to develop context-aware mobile applications. In Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia (MUM '12), article 12, ACM Press, 2012.
17. Salber, D., Anind, D., and Abowd, G. 1999. The context toolkit: Aiding the development of context-enabled applications. In: Proceedings of CHI 1999, ACM, pp. 434–441.
18. Ur, B., McManus, E., Ho, M.P.Y., and Littman, M.L. 2014. Practical trigger-action programming in the smart home. In Proceedings of CHI 2014, ACM, pp. 803–812.
19. Yang, J. and Wigdor, D. 2014. Panelrama: enabling easy specification of cross-device web applications. In Proceedings of CHI 2014, ACM, pp. 2783-2792