Beyond Responsive Design: Context-Dependent Multimodal Augmentation of Web Applications

Giuseppe Ghiani, Marco Manca, Fabio Paternò, and Claudio Porta

CNR-ISTI, HIIS Laboratory, Via G. Moruzzi 1,
56124 Pisa, Italy
{giuseppe.ghiani,marco.manca,fabio.paterno,
claudio.porta}isti.cnr.it

Abstract. Context-dependent adaptation is becoming a continuous necessity since we access our applications in more and more variegated contexts. Multimodality can be a significant support in such changing settings. We present a solution for obtaining automatic augmentation of Web applications in such a way as to enable them to exploit various combinations of graphical and vocal modalities. We report on the software architecture supporting such augmentations and its underlying context manager, as well as some example applications and first user tests.

Keywords: Adaptation, Context-Awareness, Multimodal Interfaces, Vocal Interaction, Web applications.

1 Introduction

One consequence of the explosion of mobile technology has been that we use our applications more and more in dynamic contexts of use. Such contexts can vary in aspects related to the users (their preferences, abilities, physical and emotional state, etc.), the technology (devices, connectivity, interaction modalities, etc.), environment (noise, light, etc.), and social aspects (privacy, trust, etc.).

Responsive design [14] has recently been widely adopted by many Web developers and designers since it provides support for adapting to various device features through fluid layout and stylesheets. It moreover provides the possibility of associating various visual attributes with groups of devices identified by some features detected through media queries.

However, in many cases responsive design is not enough for various reasons:

- The contextual changes that this approach is able to detect are limited to device and window resolution, and orientation, while many other aspects may vary in the context of use that can have an impact on the user experience;
- The changes that it is possible to specify are limited to hiding/showing some elements and changing the graphical attributes; no support for multimodality is provided.

The idea behind model-based approaches [3] is to have declarative descriptions that can be used to provide implementation-independent descriptions in which the purpose of each interface element is described. This approach is meaningful as is demonstrated by the fact that HTML5 has adopted it by providing more semantic tags, even if it is limited to only graphical user interfaces. Model-based approaches involve the effort of using an additional description level. This is justified by its potential support for adaptation, which is particularly effective when it involves the use of different modalities.

In this paper, we present a novel solution that addresses the limitations of responsive design. Instead of media queries we use a context manager, a software infrastructure able to detect any type of contextual change in terms of technology, users, and environments. Then, in order to better support the user, a server is able to provide for various adaptations, including augmenting the accessed Web pages in such a way as to exploit multimodality, in particular various combinations of graphical and vocal modalities. The adaptations to perform are expressed through adaptation rules in the format event/condition/actions in which the triggering events are detected by the context manager, and the action part indicates the adaptation effects that can even exploit the vocal modality still using standard HTML, CSS, and JavaScript. We have focused on the vocal modality, in various combinations with the graphical one, since it is spreading quickly thanks to rapid improvements in TTS/ASR technology (e.g., Google speech) and can support various types of input and output, including non-speech sounds. Our environment also supports vibro-tactile feedback.

In the paper, after discussion of related work, we introduce a modality-based classification of the possible adaptations that can be supported by our solution, and describe its software architecture. We provide a description of the context manager supporting it, and report on two user tests. Lastly, we draw some conclusions and provide indications for future work.

2 Related Work

One type of approach to obtaining context-dependent applications is represented by the Context Toolkit [19], which provides explicit mechanisms to handle context-dependent events. While such toolkits allow creating context-dependent applications, we provide a solution that makes any Web application context-dependent by merely requiring designers and developers (who could be different from those who created the original application) to provide only the desired adaptation rules to the adaptation engine. Thus, we focus on adaptations of Web applications, an area that has been addressed by previous work, e.g. [12] proposed a solution for automatically retargeting Web sites through the use of machine learning techniques applied to existing designs. Our support provides broader dynamic adaptation based on context dependent adaptation rules, interpreted by our adaptation server. It is worth noting that our approach does not require rewriting or manually modifying the existing applications. We instead propose a middleware that is targeted to support developers/designers who want to augment their applications with limited effort and/or to third parties needing to augment the capabilities of Web-based applications (e.g., municipalities that aim to improve Web accessibility within the smart city).

Although adaptation of Web mobile applications has been investigated, there are still many issues to solve in this area. Researchers have tried to automate the adaptation of Web pages for touch devices. W3Touch [15] is an example tool for supporting Web pages adaptation according to user interaction by using missed links and frequent zooming as indicators of layout issues, but the adaptation rules supported do not consider the use of multimodality. We aim to an architecture able to take as input existing (Web) applications and dynamically generate multimodal versions according to the context of use.

FAME [7] provides a framework to guide the development of adaptive multimodal applications. A conceptual basis for adaptive multimodal systems and a set of guidelines for driving the development process is furnished, but no support for automatic application development is given. A survey [10] about trends in context of use exploitation in mobile HCI has provided a classification of context aspects. According to the study, the most exploited aspects are related to user social dimensions and physical environments. A more recent study about mobile multimodality has led to some further guidelines for developers of context-aware adaptive multimodal mobile applications [8]. As in our vision, context-awareness is a medium for providing the combination of interaction modalities that best suit the current situation. The flexibility of our approach, as described in the following sections, lies in the extensible adaptation rules that: (1) can take into account the context of use; (2) allow adding and tuning the level of multimodality to better suit users in each run-time context. This is a type of Web augmentation, where the rendering of an existing application is augmented to improve user experience [6], different e.g. from mashup techniques, aiming to build new applications out of existing Web resources.

We enable the combination of vocal and graphical modalities both in input and output in various ways, similar to those indicated by the CARE (Complementarity, Assignment, Redundancy, and Equivalence) properties [5].

Multimodal augmentation can even support people with permanent or transient disabilities that may have problems in accessing the original Web applications, thus being a possible solution for personalised dynamic accessibility [9].

An approach to generating accessible user interfaces from multimodal design patterns is presented in [18], which requires the use of a statecharts-based notation in the authoring of the adaptive applications. Another framework for developing adaptive multimodal interfaces was introduced in [1], while our approach can be applied to any existing Web interactive application independently of how they were developed. We propose to express adaptation through rules structured in the event/condition/actions format. The event is something happening at a given time, the condition (optional) is a constraint to be satisfied, and the action describes how the application should change in order to perform the requested adaptation. A similar language for adaptation rules was used in [16] in an approach that supported adaptation as well, but without considering multimodality. Multimodal adaptation was tackled by the GUIDE project as well [4]. In that case, the authors mainly focused on adaptation for elderly people based on user capabilities models while in this work we aim to address a wider set of contextual aspects.

3 Adaptation Classification According to Modalities Use

In this section we introduce a classification of four adaptation types that illustrate the space of possible adaptations that our environment can support. In the rest of the paper we focus on the types that involve multimodality, since context-dependent multimodal augmentation is the main contribution of this work. Here we also mention graphical-to-graphical adaptation scenarios just to indicate that we are able to address them, but we do not go in depth on this part since in this paper our focus is on multimodal support.

Graphical-to-graphical adaptation. In some cases there is a need to change the user interface within the same modality. One example can be a person walking in a noisy environment. This context implies that the user is not able to pay much attention through the visual channel since s/he is walking but cannot use the vocal modality because of the noise. In this context a useful adaptation [11] can be increasing the size of the graphical elements (fonts, buttons, etc.) in order to make them more easily perceivable and selectable.

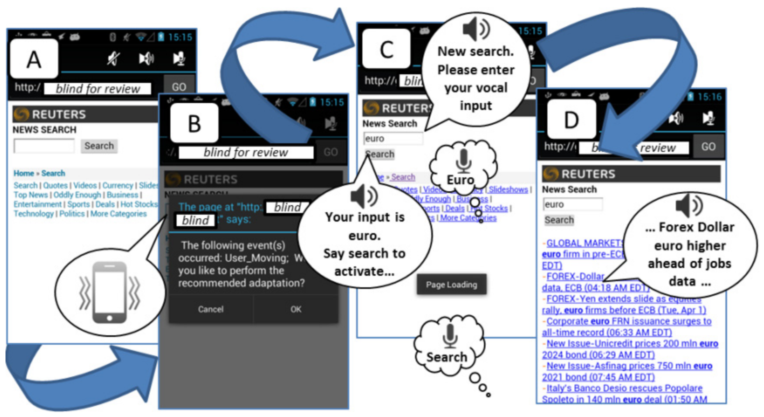
A
B
C
D
New search. Please enter your vocal input
Euro
Your input is euro. Say search to activate...
Search
... Forex Dollar euro higher ahead of jobs data ...
The following event(s) occurred: User_Moving; W you like to perform the recommended adaptation?
Cancel
OK
Page Loading

Fig. 1. An example scenario for graphical-to-multimodal adaptation

Graphical-to-multimodal adaptation. There are cases in which exploiting other modalities can allow users to overcome limitations inherent in the graphical one. For example, when the context of use limits the degree of attention to the visual modality, e.g. walking on a crowded street or while driving, such limitation can be counterbalanced by exploiting the vocal modality for supporting information rendering and interaction. Thus, some information can be rendered in a redundant way in order to ensure that it is perceived by the user, while the input can be entered either through one modality or another. This is the type of scenario that has received limited support so far, and the main contribution of this work is to provide a solution that works even for existing Web applications and not only for those developed with specific authoring tools. An example is shown in Figure 1. A graphical Web application is accessed (A) when the user is sitting. When the user starts an activity that makes full graphical interaction difficult (e.g., walking), the Platform detects the context change, triggers a vibrating alert, and asks for confirmation to perform the adaptation (B) before providing the appropriate multimodal version (C). The graphical page is thus augmented with vocal input and output. At the beginning a vocal prompt is provided ("New search. Please enter your vocal input"). The user enters "euro" vocally and confirms. Then, the multimodal version of the search results (D) is rendered. When the user skips the search form through the next command, the text content of the page is rendered vocally.

Multimodal-to-graphical adaptation. If for some reason multiple modalities can no longer be exploited at the same time, then it can be useful to change the user interface in such a way as to exploit only the graphical modality. This type of adaptation is not particularly complex since it is mainly based on disabling one modality and forcing all information rendering and interaction to be performed graphically. Thus, when dealing with a multimodal user interface that was originally graphical, the multimodal-to-graphical adaptation actually consists on restoring the original interface version.

Multimodal-to-multimodal adaptation. It may happen that once the original graphical application has been augmented in terms of multimodality due to some contextual change, at some point a new contextual change may require different multimodal support. Indeed, even within multimodal support there can be various possibilities depending on how the modalities involved are exploited. For example, in the case of low multimodality for interactive elements, the input can be entered either vocally or graphically, while prompts and feedback are rendered only graphically. On the other hand, in the case of high multimodality all interactions are rendered through both modalities. Likewise, output textual elements can be rendered vocally in different ways: totally or partially or only on request depending on the parameters specified in the adaptation rules.

4 Software Architecture of the Adaptation Platform

Our platform for multimodal augmentation is server-based. It includes a proxy and an adaptation server composed of a number of modules, which also use model-based languages for supporting adaptation. In such languages it is possible to have descriptions at various abstraction levels. We have considered the use of abstract and concrete descriptions. In the former the specification is independent of the actual interaction modalities used, while in the latter the description depends on such interaction modalities but it is still independent of the implementation languages used. We have adopted the MARIA language [17] since it provides both an abstract language and concrete refinements of such language for various platforms (graphical, vocal, multimodal, etc.) and it is publicly available along with an authoring environment. In the following we first introduce the various modules (graphically represented in Figure 2) and then describe how they communicate with each other.

Orchestrator. It is mainly the interface between the adaptation server and the external world. It coordinates the access to and interactions among the various modules of the adaptation server.

Navigation Proxy. In addition to the usual proxy functionalities the proxy that we have implemented inserts some scripts in the navigated pages in order to allow communication between such pages and the orchestrator module. These scripts are used to trigger and perform the forwarding of the DOM of the currently accessed page to the server, and to dynamically load the adapted page.

Reverser. This tool is able to create the MARIA graphical concrete description of any existing Web page. For this purpose, it parses the page DOM, saves the content of the script nodes in a separate file, adds the information related to CSSs to a cache memory, including information related to elements created dynamically through JavaScript. Then, it starts a depth-first analysis of the DOM tree nodes (except when there are elements that can be associated with labels; in this case the labels are searched through a breadth-first analysis). For each node the reverse engineering tool identifies the type, the relevant CSSs and events. The analysis considers CSSs that can be external to the Web page, internal to the page with the STYLE element, and in any HTML element with the *style* attribute, including the case of nested CSSs. Then, it creates the corresponding elements, with the associated attributes and events, in the logical concrete descriptions. In contrast to previous reverse engineering tools [2], this module is also able to handle recently introduced HTML 5 and CSS 3 elements.

Adaptation Engine. The purpose of this module is to decide what adaptation should take place. For this purpose it uses adaptation rules written in an XML-based format according to the event/condition/action template. When new rules are added the adaptation engine subscribes to be notified of their events by the context manager. In this way when the relevant events occur the associated actions are triggered. Such actions can vary from small changes (e.g. change of font size) to the change of the platform used for rendering the user interface (as in the case of graphical-to-multimodal adaptation).

Multimodal Adapter. The multimodal adapter performs the transformation from the graphical concrete description to the multimodal one. Since both are XML-based descriptions it has been implemented as an XSLT transformation. The action part of the adaptation rules can specify the parameters for this transformation. For instance, it is possible to indicate if the images must be vocally annotated (i.e. by synthesizing the “alt” attribute), if users have to confirm the entered vocal input, if the multimodal support has to list all the choice elements in case of a single choice interactor. The classification of the interactive elements provided by the MARIA language together with the CARE properties provide a powerful way to customise the multimodality level. Since the user interface specification has a hierarchical structure, it is possible, through the CARE properties, to associate the type of desired multimodality at different granularity levels. It is also possible to associate a multimodality property to all the elements that share a certain semantics (e.g. single selection object, text edit, function activation) according to the classification of the abstract MARIA language. In the MARIA multimodal Concrete User Interface (CUI), the interaction elements can be decomposed into three different parts: input, prompt and feedback. In the adaptations rules it is possible to indicate the CARE value for each part, thus specifying how users can interact with interaction elements with a very fine level of granularity. Usually the Multimodal Adapter extracts vocal information from the graphical page, for instance the vocal prompt value is derived from the label text of the interactor element. If the element has no label then the multimodal adapter adds a default vocal prompt. Regarding text content, the way to vocally render it can be specified as well (e.g., synthesize the whole text or only the heading part, synthesize it automatically or when selecting it, etc.).

Context Manager. It supports the possibility of detecting any change in the context of use. It has a client/server architecture in which the clients are context delegates whose task is to detect various types of contextual events and inform the server, which will then communicate them to the adaptation server, if it had subscribed for their notification. A more detailed description of this important component follows in the next section.

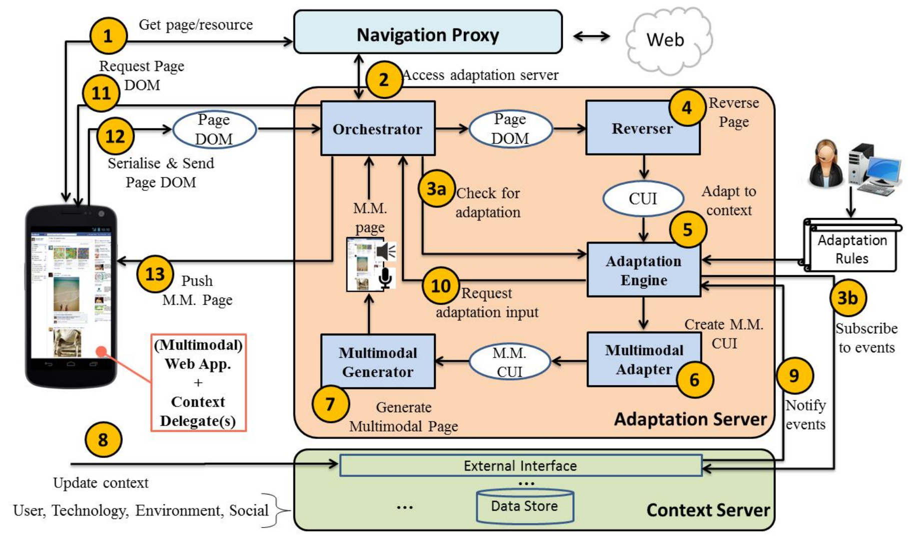

Fig. 2. The architecture of the proposed environment

Multimodal Generator. The adaptation server also exploits a generator of multimodal interfaces from model-based descriptions, which was introduced in [13]. That work, however, relied on existing multimodal model-based descriptions of the interactive Web application, and was not able to support adaptation of existing Web applications, as presented in this paper.

Communication among components. Figure 2 shows the steps through which the multimodal augmentation takes place (the numbers in Figure 2 provide an indication of the corresponding step):

1) The Web application is accessed through the Proxy, which enhances it (e.g. by modifying links/references, injecting additional scripts). The user interacts with the modified Web application in the same way as without any proxy.
2) The Proxy passes the annotated page DOM to the Orchestrator module of the Adaptation Server.
3) a) The Orchestrator queries the Adaptation Engine to check whether an adaptation is needed for the current page (i.e. whether, according to the adaptation rules, the page should be adapted to the current context). If adaptation is needed, then the Orchestrator sends the page DOM to the Reverser; otherwise, the Proxy receives back the current page and provides it to the client browser without any change.
   b) At the first user access the Adaptation Engine subscribes to the relevant context events according to the available adaptation rules associated with the current user and application.
4) The Reverser creates the concrete description (CUI) and provides it to the Adaptation Engine.
5) The Adaptation Engine identifies what adaptation should take place according to the current adaptation rules.
6) If deemed useful, the Multimodal Adapter is invoked to convert the graphical concrete description to a multimodal concrete description.
7) The Multimodal Generator takes such multimodal concrete description (MM CUI) and creates the actual multimodal page implementation, which is passed to the Orchestrator, which in turn provides it to the Proxy.
8) If at some point something changes in the context of use (e.g., the user has started walking fast), the corresponding Context Delegate informs the Context Server.
9) If the Adaptation Engine had previously subscribed to the event that has occurred, then the Context Server notifies it.
10) The Adaptation Engine asks the Orchestrator for the adaptation input (i.e. the current page DOM).
11) The Orchestrator requests the current page DOM from the client browser.
12) The client browser serializes the current Web page DOM (this is managed by scripts injected by the Proxy) and sends it to the Orchestrator, then steps 4 to 7 are performed again.
13) The Orchestrator pushes the current multimodal page to the client browser.

5 Context Information Management

The context manager is composed of a context server and a number of context delegates and it has the aim of detecting, communicating, and manipulating context information. In particular, the context delegates are modules devoted to detecting contextual information. A context delegate, upon sensing several context parameters, updates the corresponding attributes in the context server. Context delegates can be deployed in the user device or in the environment.

The context manager supports contextual information manipulation at various abstraction levels. Delegates and server exploit a REST interface, which allows delegates to update various entities in the server even without knowing their structures exactly. Some of the RESTful services defined so far are: UserEnvironment, which includes attributes such as noise/light/temperature; UserLocationBluetoothBeacon, for the state (e.g. signal strength) of a Bluetooth beacon detected by the device; User-PhysiologicalActivity, for user respiration and heart rate.

In order to automatically update the previously mentioned resources, we have implemented a number of context delegates for Android devices. One example senses Bluetooth beacons (e.g. Bluetooth embedded devices, dongles, etc.) available in the environment and provides the context server with their identifier (name, MAC address) and Received Signal Strength Indicator (RSSI). Two other different delegates are devoted to updating the user environment: one for the light and one for the noise level. The former parameter is provided by the device light sensor; the latter is obtained by analysing the audio input amplitude provided by the embedded microphone. A context delegate has also been developed for interfacing with physiological monitoring hardware with Bluetooth interface. The physiological monitor is a belt that can be comfortably worn and senses several parameters (e.g. heart rate, ECG-variability, respiration rate, user movements). The above cited context delegates and related data are exploited at run time depending on the adaptation rules.

Besides the basic operations for creating, updating and removing information, the Context Manager also hosts mechanisms for subscribing to information changes. Subscription is a way for an external module to be asynchronously notified when something changes in the context, and thus it is fundamental for the multimodal context-aware adaptation platform. We use an Event subscription mechanism, i.e. a mechanism for a subscriber to be notified only when one or more conditions are met. Conditions can refer to one or more attributes. The attributes are identified through a path in their hierarchical structure that represents the context model (e.g., the noise level attribute is defined as an environment sub-entity). Notifications are sent via HTTP POST. Subscriptions are carried out at run-time according to pre-defined adaptation rules structured as event-conditions-actions, as previously discussed.

We designed the context manager to be able to detect any relevant contextual information: in the server a lower layer stores and manages information as a set of structured entities, while the external interface allows access to such information at different abstraction levels. The benefit of the low level way of storing information is flexibility. Low level entities are basic data containers, able to store an arbitrary number of attributes and/or reference to other entities (e.g. user, device, etc.). Overall, the context manager architecture is extensible, since it is easy to add new delegates for detecting further contextual events without having to change the implementation of the other modules.

6 User Tests

We have carried out two user tests in order to assess some aspects characterizing context-based multimodal augmentation and to what extent they are considered useful and usable with respect to the standard way to access Web applications.

A first user test involved ten users (6 male, 4 female) between 21 and 45 y.o. (mean 30.4), who were requested to interact with the English mobile Wikipedia home page (http://en.m.wikipedia.org) in order to find information about cities and countries. Only four users had previous experience in the use of UIs combining graphical and vocal modalities. None of them was involved in the project. Interaction occurred through an Android smartphone equipped with support for multimodal Web user interface execution obtained through an instance of a WebView component and libraries accessing the Google Text-To-Speech and Augmented Speech Recognition. The calls to the functionalities of such libraries are created by the generator for MARIA multimodal concrete descriptions.

In the test, the users first had to search for a city, and find out its population, and then to search for a country and find its surface area. The city and country names were the same for all users, in order to maintain homogeneity in task complexity.

Users had to complete the tasks through a multimodal augmented version of Wikipedia automatically obtained through our adaptation environment according to the context of use. When a contextual change triggers an adaptation various events happen on the device during this adaptation transition: a pop-up announces the contextual change and the possibility of triggering multimodal adaptation; if accepted, there is vibrotactile feedback and a pop-up informing the user of the progression of the adaptation transition in percentage. The contextual factor for triggering the adaptation was the environment noise, detected by a context delegate running in background on the smartphone.

In order to evaluate our platform, we defined a list of relevant aspects related to usability and usefulness. Such aspects, listed in the following, were rated with a value on a 1-5 scale, with 5 as the most positive score (min and max value are expressed into square brackets):

a) Awareness of context-dependent interface adaptation [3,5]; mean: 4.1; std.: 0.88;
b) Adaptation appropriateness [1,4]; mean: 3.4; std.: 0.97;
c) Adaptation continuity [1,5]; mean: 3.2; std.: 1.03;
d) Rendering of adaptation transition [1,5]; mean: 2.4; std.: 1.35;
e) Impact of the adaptation in decreasing interaction complexity [1,5]; mean: 3.2; std.: 1.03;
f) Impact of adaptation in improving user experience [1,5]; mean: 3.2; std.: 1.03;
g) Utility of multimodal augmentation for improving Web applications usability [2,5]; mean: 3.7; std.: 0.82.

In order to help users in subjectively rating the system, each possible score in the questionnaire was associated to a short textual description. For instance, in the first question, 1/2/3/4/5 values on the Scale were described as "*My awareness was very low/low/borderline/high/very high*", respectively.

Overall, we can argue that users were quite aware of the context-dependent interface adaptation being performed, in particular thanks to the vibrotactile notification following the contextual change. Other issues emerged on adaptation appropriateness, which received lower ratings. This may be due to minor inconsistencies the users found in the adapted page (e.g. links not immediately visible because of their altered layout).

The way adaptation transition was rendered received the lowest mean score. A possible explanation for this is the server-side delay, i.e. page adaptation taking much longer than simple loading.

Among the positive informal considerations, participants highlighted the benefits that the multimodality can provide in various situations, such as when hands free interaction is needed, and/or when it is not possible/safe to look at the screen. One participant also mentioned possible benefits in the social aspects of interaction that can arise from such multimodal adaptations (e.g., accessibility for the visually impaired). We also got some positive observations on the potential of multimodal adaptation, for which our platform was considered to be a good starting point.

In general, the main technical issue with the multimodal augmented version seemed to be the latency of the adaptation process. The preliminary results collected and the issues pointed out by the participants helped us to create an improved version of the platform. In particular, the adaptation process latency, which was considered a serious lack, was reduced by reducing the computation complexity (especially for the Reverser module) and by installing all the modules in a single server, which led to minimal network communications.

In December 2013 we performed a second user test taking into account the indications and issues pointed out by the users during the first test. We decided to assess the same aspects as in the previous test, thus the assessment questionnaire was similar, though we added one more question about the adaptation responsiveness.

Twelve users participated in the second user test (8 male, 4 female), with ages ranging between 21 and 36 (mean 30). Users were recruited from within the personnel of our organization was among those who were not involved in the project. They participated voluntarily and had to perform three tasks for different applications.

Walking with noise scenario: the users had to walk in a noisy scenario and interact with the original (i.e. not adapted) Web site at http://m.foxbusiness.com. After entering a search into the form and pressing the submit button, the context changed (the noise level decreased) and the adaptation process started. Users then had to insert a different text input and listen to the voice synthesis of the search results. In the resulting multimodal version we had: redundant (vocal and graphical) text output, and only graphical input/prompt/feedback of interaction elements.

Fitness scenario: the user was doing gymnastics. Initially s/he accessed the original Web site at http://us.mobile.reuters.com/search and performed a search through the non-adapted page. During an exercise s/he had to change position from standing to lying down and had to continue the interaction with the opened page. When the user was laying down, the physiological delegate running on the device was able to interpret this change and trigger the page adaptation. The user could then insert the input vocally and listen to the voice synthesis of the search results. In the resulting multimodal version we had: redundant (vocal and graphical) text output, while in the interaction elements the input was equivalent, prompt and feedback were redundant, and the ask for confirmation parameter was false. Because the user was able to see the entered vocal input graphically, we decided not to ask for vocal confirmation of the input.

User driving scenario: we simulated a car through a large screen showing a street view video and users had a steering wheel. The user started this scenario outside the room (outside the car) and performed a search in the original Web site at http://eurometeo.mobi.en. When the user entered the room and sat at the desk with the large screen and wheel, the adaptation took place. The context manager detected when the user was inside the car thanks to a context delegate for Bluetooth beacons. When the Bluetooth beacon with the strongest signal was the car's, then the adaptation was triggered. In this scenario users could insert the input only vocally (the input field was not visible) and the output was redundant (graphical and vocal). In this way, users could interact with the interface while driving without taking their hands off the wheel. In the resulting multimodal version we had: redundant (vocal and graphical) text output, input/prompt/feedback of interaction elements only vocal, ask for confirmation true. In this scenario the input prompt and feedback part of the interaction elements (text edit and submit button) were only vocal. This means that they were not visible but it was possible to interact with them vocally; for this reason and in order not to distract the user while driving we decided to ask the user to confirm the entered input.

Therefore, for each task, users interacted with the original, non-adapted Web site through the graphical modality to perform a search and to read the results. After that, the context changed and the adapted page was loaded, and users had to perform the same task, with different search input and different result output, through the adapted page.

The following aspects were rated (Scale and values as for the previous test):

a) Awareness of context-dependent interface adaptation [4,5]; mean: 4.3; std.: 0.49;
b) Adaptation appropriateness [3,5]; mean: 4.08; std.: 0.52;
c) Adaptation continuity [3,5]; mean: 4.17; std.: 0.58;
d) Rendering of adaptation transition [3,5]; mean: 4.0; std.: 0.74;
e) Impact of the adaptation in decreasing interaction complexity [1,5]; mean: 3.58; std.: 1.16;
f) Impact of adaptation in improving user experience [3,5]; mean: 4,17; std.: 0.58;
g) Utility of multimodal augmentation for improving Web applications usability [3,5]; mean: 4.0; std.: 0.74;
h) Adaptation responsiveness [3,5]; mean: 3.92; std.: 0.67.

All aspects received greater ratings than in the previous test (excluding *h*, absent in the first test).

The rating on Awareness of the Adaptation process is slightly higher than in the first test. This is presumably because with respect to the previous version of the system, the vibrotactile notification has been combined with a pop-up also including information about the event that triggered the adaptation.

The Adaptation Appropriateness received good ratings, higher than the first user test. We believe this improvement is due to the new version of the multimodal generator, which has been updated in order to solve minor layout issues mentioned by the users of the first test.

The Adaptation Continuity and the Rendering of Adaptation transition received significantly higher ratings than previously (especially the latter). This is likely due to the improvements to the system that reduced the adaptation process latency. This is confirmed also by the good ratings of the adaptation responsiveness.

The Impact of the adaptation in decreasing interaction complexity received borderline ratings (nearly the same rating as the first test). This seems to be due to the accented English pronunciation of the participants. The problem of erroneous vocal input recognition is not due to the multimodal adaptation system, but is influenced by the distance from the microphone, the loudness of the voice and user pronunciation.

Two users did not like the need to press OK to confirm the adaptation when the pop-up appears because it forced them to manually interact with the interface even though the system adapted the page to interact vocally.

We have logged the interaction times for each scenario performed by each user. Scenario A (walking with noise) took between 97 and 490 seconds (mean 245, std. 132). For scenario B (fitness), interaction times were between 110 and 410 seconds (mean 198, std. 96). Completion times for scenario C (driving) was between 120 and 308 (mean 185, std. 52). Although a proper time comparison across modalities would have implied to perform the same task in the same settings, it is interesting to observe that scenario A, which took longer on average, relied on only-graphical input. Scenarios B and C (with multimodal and only-vocal input, respectively) took less time. Since the scenarios order was shuffled for the various users, we would exclude that this is due to a learning effect. In particular, scenario C, where vocal interaction was dominant, was the one whose tasks were performed most quickly. It is also worth highlighting that the standard deviation for interaction times was lower for scenario C than for B. In addition, scenario B had, in turn, a lower standard deviation than A (the one without vocal input). This may indicate that vocal interaction, which already brings undeniable benefits (e.g. accessibility) does not negatively affect task completion time.

The other statistical tests we have performed, e.g. considering ratings on system responsiveness and on perceived improvement in user experience, did not highlight any significant correlation between task completion time and subjective ratings.

We plan to perform a more extensive user study in the future by involving a larger set of users in order to get more statistically significant information.

7 Possible Platform Deployment

In order to highlight the benefits of the proposed approach for augmenting Web applications, in this section we explain how our platform can be deployed and exploited. To this end, there are two main modes depending on the organizations involved.

In within-organization deployment the adaptation platform can be deployed in and managed by the same organization that hosts the Web application(s) to be augmented. In this case, those responsible for configuring the platform are internal to the organization, e.g., the developers/designers responsible for managing the Web applications. It is worth noting that the existing Web applications do not need manual modifications or upgrades to be integrated with the adaptation platform.

In cross-organization deployment the organization hosting and managing the adaptation platform can be different from the one providing the original Web applications. An example use case is represented by a company or institution needing to provide (a subset of) users with augmented versions of existing Web interfaces with minimal effort. Examples might be a municipality wishing to furnish multimodal access to tourist information; a transportation company needing to provide its drivers with accessible versions of a pre-existing application for logistics, etc.

It should be noted that, in both deployments, end users access the Web applications through the (proxy of the) Augmentation Platform.

The formal issues arising from this kind of deployment, such as the need for agreements between the two organizations involved to comply with local/international rules, are beyond the scope of this paper.

8 Conclusions

We have presented a novel solution for multimodal augmentation of Web applications in mobile scenarios. It exploits an adaptation server receiving events from a distributed context manager and able to trigger adaptations according to externally specified rules. The server also exploits a model-based description language to perform the transformation from a graphical to a multimodal concrete description.

A video showing example applications of our platform for multimodal augmentation is available at: http://youtu.be/7Y670aWNUDM .

We have reported on user tests, which gave positive feedback and indications for further refinements. The proposed approach provides more flexible and general support than that currently provided in responsive design, which is not able to address many types of contextual events or provide dynamic and customizable multimodality.

Future work will be dedicated to making the adaptation server publicly available for external use, improve its performance, and move some of its functionality to cloud support. Since creating/editing adaptation rules still require some basic technological knowledge, we also aim to provide end users with intuitive interfaces for the customization of such rules. This would allow them to directly modify the context-dependent behaviour of their Web applications also in terms of multimodality.

References

1. Avouac, P.-A., Lalanda, P., Nigay, L.: Autonomic management of multimodal interaction: DynaMo in action. In: Proc. EICS 2012, pp. 35–44. ACM (2012)

2. Bellucci, F., Ghiani, G., Paternò, F., Porta, C.: Automatic Reverse Engineering of Interactive Dynamic Web Applications to Support Adaptation across Platforms. In: Proc. IUI 2012, pp. 217–226. ACM (2012)
3. Cantera, J.M., González, J., Meixner, G., Paternò, F., Pullmann, J., Raggett, D., Schwabe, D., Vanderdonckt, J.: Model-Based UI XG Final Report, `http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/`
4. Coelho, J., Duarte, C.: The Contribution of Multimodal Adaptation Techniques to the GUIDE Interface. In: Stephanidis, C. (ed.) Universal Access in HCI, Part I, HCII 2011. LNCS, vol. 6765, pp. 337–346. Springer, Heidelberg (2011)
5. Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R.: Four Easy Pieces for Assessing the Usability of Multimodal Interaction: the CARE Properties. In: Proc. INTERACT 1995, pp. 115–120. Chapman and Hall (1995)
6. Díaz, O., Arellano, C., Azanza, M.: A language for end-user web augmentation: Caring for producers and consumers alike. ACM Transactions on Web (2), 9 (2013)
7. Duarte, C., Carriço, L.: A conceptual framework for developing adaptive multimodal applications. In: Proc. IUI 2006, pp. 132–139. ACM (2006)
8. Dumas, B., Solórzano, M., Signer, B.: Design Guidelines for Adaptive Multimodal Mobile Input Solutions. In: Proc. MobileHCI 2013, pp. 285–294. ACM (2013)
9. Gajos, K.Z., Hurst, A., Findlater, L.: Personalized dynamic accessibility. Interactions 19(2), 69–73 (2012)
10. Jumisko-Pyykkö, S., Vainio, T.: Framing the Context of Use for Mobile HCI. International Journal of Mobile-Human-Computer-Interaction (IJMHCI) 3(4), 1–28 (2010)
11. Kane, S.K., Wobbrock, J.O., Smith, I.E.: Getting off the treadmill: Evaluating walking user interfaces for mobile devices in public spaces. In: Proc. MobileHCI 2008, pp. 109–118. ACM Press (2008)
12. Kumar, R., Talton, J.O., Ahmad, S., Klemmer, S.R.: Bricolage: Example-Based Retargeting for Web Design. In: Proc. CHI 2011, pp. 2197–2206. ACM (2011)
13. Manca, M., Paternò, F., Santoro, C., Spano, L.D.: Generation of Multi-Device Adaptive MultiModal Web Applications. In: Daniel, F., Papadopoulos, G.A., Thiran, P. (eds.) MobiWIS 2013. LNCS, vol. 8093, pp. 218–232. Springer, Heidelberg (2013)
14. Marcotte, E.: Responsive Web Design, A Book Apart (2011), `http://www.abookapart.com/products/responsive-web-design`
15. Nebeling, M., Speicher, M., Norrie, M.C.: W3Touch: Metrics-based Web Page Adaptation for Touch. In: Proc. CHI 2013, pp. 2311–2320. ACM (2013)
16. Octavia, J.R., Vanacken, L., Raymaekers, C., Coninx, K., Flerackers, E.: Facilitating Adaptation in Virtual Environments Using a Context-Aware Model-Based Design Process. In: England, D., Palanque, P., Vanderdonckt, J., Wild, P.J. (eds.) TAMODIA 2009. LNCS, vol. 5963, pp. 58–71. Springer, Heidelberg (2010)
17. Paternò, F., Santoro, C., Spano, L.D.: MARIA: A Universal Language for Service-Oriented Applications in Ubiquitous Environment. ACM Transactions on Computer-Human Interaction 16(4), 19:1–19:30 (2009)
18. Peissner, M., Häbe, D., Janssen, D., Sellner, T.: MyUI: generating accessible user interfaces from multimodal design patterns. In: Proc. EICS 2012, pp. 81–90. ACM Press (2012)
19. Salber, D., Anind, D., Abowd, G.: The context toolkit: Aiding the development of context-enabled applications. In: Proc. CHI 1999 Conference, pp. 434–441. ACM (1999)