Supporting Adaptation of Web Applications to the Mobile Environment with Automated Usability Evaluation

Luiz F. Gonçalves
POSCOMP
Federal University of Itajubá
luiz.felipe.lfg@gmail.com

Leandro G. Vasconcelos
National Institute for Space Research
leandro.guarino@lit.inpe.br

Ethan V. Munson
Department of EECS
University of Wisconsin-Milwaukee
munson@uwm.edu

Laércio A. Baldochi
Institute of Mathematics and Computing
Federal University of Itajubá
baldochi@unifei.edu.br

ABSTRACT

The year 2014 marked an important shift in the Web's history, as users started to spend more time surfing the web using mobile devices than using desktop computers. However, today, a large proportion of websites are still not designed for good mobile device access. To tackle this problem, this paper proposes a new approach to adapting desktop-based web applications to the mobile environment. Our approach is supported by a task-based usability evaluation tool called MOBILICS, which is able to evaluate desktop-based web applications used in mobile devices. Based on the evaluation results, our tool provides detailed recommendations for fixing the detected usability problems. We conducted an experiment which demonstrates that our approach provides useful support for the adaptation of a desktop-based web application into a mobile version.

CCS Concepts

•**Human Computer Interaction** → **Usability Testing;** *Web Interfaces;*

Keywords

Usability evaluation, touch interaction, desktop-to-mobile adaptation

1. INTRODUCTION

The use of mobile platforms for web access is growing at a fast pace so that, today, more than half of web users access the web via a mobile device [5]. The problem is that a significant proportion of websites are not designed to match the affordances of mobile devices. In spite of this, user expectations for mobile usage are high: they expect that web applications will load just as fast as they do on desktops. In fact, Hof [8] reports that a third of users surveyed said they were likely to go to a competitor's site if they got annoyed while using a web application ill-suited to mobile access.

This scenario clearly shows that companies need to provide mobile-friendly web applicaitons in order to avoid financial losses. However, developing different versions of a web application, or a single web application that supports both desktop and mobile access, is costly and time consuming. In order to tackle this problem, researchers have proposed different approaches for automatically adapting desktop-based web applications to small form factor devices.

The first attempts to adapt web pages to small screens are Digestor [2] and Power Browser [6]. By applying text summarization, these tools build summary pages containing links to different parts of the original pages. Each part is presented by itself, using the entire mobile device's display. Another approach to automatically adapting Web pages is page fragmentation [7]. This technique breaks a web page into fragments that can fit in the mobile device screen and creates a two level hierarchy to navigate within the page.

There are other approaches that exploit scaling to better use the limited screen space on mobile devices. Minimap [13], for instance, scales down non-textual elements such as images while maintaining the size of textual elements. Following Minimap's principle that some parts of the page are irrelevant, Baudisch et. al. [1] proposed Collapse-to-Zoom, a tool that allows users to collapse the content of a page that is not interesting, allowing the remainder of the page to expand.

In spite of these attempts, automatic adaptation of web pages generally has undesirable side effects. Both summarization and fragmentation techniques require the user to move back and forth between a main summary page and smaller content pages, which requires many actions from the

SAC 2016,April 04-08, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2851613.2851863

user and may lead to loss of context. On the other hand, approaches based on content relevance, such as Minimap, require a means of determining what is irrelevant — an image may be irrelevant to some, but relevant to others.

A drawback common to all these approaches is the fact that, no matter how different two desktop web applications are, the resulting, automatically adapted, mobile versions of the two applications will function very similarly. This is not interesting for business, because companies need to provide exclusive features to differentiate their websites from their competitors. For this reason, businesses generally develop mobile web applications from scratch, following design guidelines for small form factor devices [12] and, more recently, applying responsive design approaches, such as those provided by Bootstrap (bootstrap.com) and WebFlow (webflow.com).

Developing a mobile web application from scratch, however, may throw away all the effort that has been spent on the development of a desktop version. We think that, at least, some of this effort can be reused. In other words, we argue that the desktop version of a web application can be a useful starting point for the development of the mobile version. For example, while some pages may require a complete rebuild, others may only need small fixes in order to function well and a few may even be used as is.

In order to assist developers adapting desktop websites to the mobile environment, we developed MOBILICS, an automatic usability evaluation system for mobile web applications. MOBILICS is an extension to USABILICS [14, 15], which performs automated task-based usability evaluation of conventional, desktop web applications.

Our approach to assist the adaptation of web applications works in the following way: we define tasks for a given desktop web application using a desktop machine. Following, we capture the interactions of end users performing these tasks using mobile devices. The captured tasks are evaluated, *i.e.*, compared to the previously defined tasks. Based on this evaluation, MOBILICS detects usability issues, pointing out the tasks that presented problems and the interface elements for which these problems were detected. Moreover, it provides recommendations for improving the application's mobile usability.

Experiments performed with MOBILICS show that this approach is effective in assisting developers to adaptat desktop web applications to the mobile environment. By following our approach, developers are able to build a mobile web application reusing a significant amount of code from its equivalent desktop-based application, thus saving substantial development effort. Moreover, the resulting mobile web application retains the overall look and feel of the original application.

This paper is organized as follows. Section 2 presents our previous work on usability evaluation. Section 3 presents the extensions we made in order to support the usability evaluation for the mobile environment. Section 4 evaluates our approach to support the generation of a mobile website from a desktop website. Section 5 compares our approach to related work. Finally, Section 6 presents our final remarks.

2. PREVIOUS WORK

The World Wide Web presents a clear structural pattern in which websites are composed of a collection of pages that, in turn, are composed of elements such as hyperlinks, tables, forms, etc, which are usually grouped by special elements such as DIV and SPAN. By exploiting this pattern, and considering that interface elements are usually shared among several pages, we proposed COP [14], an interface model that aims at facilitating the definition of tasks.

The main concepts in COP are ***C****ontainer*, ***O****bject* and ***P****age*. An object is any page element that the user may interact with, such as hyperlinks, text fields, images, buttons, etc. A container is any page element that contains one or more objects. Finally, a page is an interface that contains one or more containers.

Besides exploiting the fact that containers and objects may appear in several pages, the COP model also exploits the similarities of objects and containers within a single page. In any given *page*, an object may be unique (using its id) or similar to other objects in terms of formatting (i.e. border or font type, color, etc.) and/or in terms of content (i.e.: texts and images). The same applies to containers: a container may be identified in a unique way, or it may be classified as similar to other containers, but only in terms of formatting.

The COP model was the foundation for the development of USABILICS, a task-oriented remote usability evaluation system. USABILICS evaluates the execution of tasks by calculating the similarity among the sequence of events produced by users and those previously captured by evaluators. By using USABILICS, evaluators may benefit from the COP model to define generic tasks, thus saving time and effort to evaluate tasks. The approach provided by USABILICS is composed by four main activities: (i) task definition, (ii) logging, (iii) task analysis, and (iv) recommendations.

i) Task definition. One of the goals of USABILICS is to perform usability evaluation with a minimum burden on the application developer. To achieve this, we implemented UsaTasker [16], a task definition tool that allows developers to define tasks by simply interacting with the application's GUI. UsaTasker provides a user-friendly interface for the management of tasks, where the evaluator can create (record), view, update and delete tasks. For recording a task, all that is required is to use the application as it is expected from the end user. While the evaluator surfs the application interface, she is prompted with generalization/specialization options, as specified by the COP model.

ii) Logging. USABILICS embeds in Web pages a Javascript client application that recognizes all page elements using the Document Object Model (DOM) and binds events to these elements, allowing the gathering of user interactions such as mouse movements, scrolling, window resizing, among others. Events generated by the pages of the application, such as *load* and *unload* are also captured. Periodically, the client application compresses the logs and send them to a server application, which stores the data in a relational database for being used during the task analysis phase.

iii) Task analysis. USABILICS performs task analysis by comparing the sequence of events recorded for a given

task and the corresponding sequence captured from the end users' interactions. The similarity between these sequences provides a metric of efficiency. The percentage of completeness of a task provides a metric of effectiveness. Based on these parameters, we proposed a metric for evaluating the usability of tasks called the *usability index* [15]. In order to compare two sequences of events, USABILICS calculates the similarity of each subsequence identified in the end user interaction. The identification of the set of subsequences that match a given task is performed comparing the generalization options defined in the COP model applied both to events produced by the end user interaction and to events recorded during the definition of the correspondent task.

iv) Recommendations. When comparing the sequences of events that compose a task it is possible to identify three different situations that indicate the occurrence of incorrect actions: (i) an action does not belong in the correct sequence of actions, (ii) an action is omitted from the sequence, (iii) an action within the sequence is replaced by another action. USABILICS is able to identify these three types of wrong actions and the interface components associated to them. By analyzing a set of different tasks presenting low usability, we found out that wrong actions are mainly related to hyperlink clicks, to the opening of pages, to the scrolling in pages and to the interaction with forms. We defined 6 recommendations for fixing these issues. An experiment showed that, by following our recommendations, developers were able to improve the usability of applications significantly [15].

3. MOBILICS

In recent years the web has become the universal medium for the development of software applications. Today, web developers can count on a wide variety of tools and frameworks that aid the development of desktop web applications. Desktop web browsers have also evolved significantly since the inception of the web. Today, they are robust and reliable pieces of software and, by adhering to W3C standards, they simplify the creation of cross browser applications.

When compared to desktop versions, mobile web applications are still in their infancy. Although there are tools and frameworks that aid the development of mobile websites, mobile browsers are not yet mature, making cross-browser development very difficult. In addition, there are also cross-device and cross-platform issues. As a result, the behavior of mobile web applications may vary substantially among browsers, platforms and even mobile devices, making it complex to evaluate their usability. Therefore, a substantial amount of work is required to address these compatibility issues. The following subsections present the extensions made in USABILICS in order to implement the MOBILICS system and discuss how we dealt with these issues.

3.1 Touch Events

Other than the size of the display, the main difference between desktop and mobile applications is the input method. While the first is mouse-oriented, the second is based on directly touching the screen. Moreover, a touch is not equivalent to a mouse click, as there are different ways of touching the screen (*tap*, *doubletap*, *swipe*, *pinch*, *drag*, etc). As a result, instrumentation for touch events had to be implemented from scratch in MOBILICS.

One important issue regarding the capture of events is the fact that the Javascript client application in charge of the capture may not impose an overhead to the loading of pages. Thus, using third-party libraries for capturing events was not an option, as it would impact the application performance, affecting the loading time of the web application. This is the reason why we mentioned that we had to implement the touch events from scratch.

The main challenge when implementing these events is how to correctly detect them. This is because the basic events that are triggered by the listeners are only three: *touchstart*, *touchmove* and *touchend*. Therefore, to determine if an interaction is a tap, a drag, a pinch or other, it is necessary to analyze the sequence of basic events. Thus, a pinch event, for instance, is detected when two simultaneous *touchstart* events occur, followed by two *touchmove* events. It is also necessary to analyze if the user is moving the fingers closer together or further apart from each other in order to detect if it is a *zoom in* or a *zoom out*.

Therefore, the Javascript client application, used both to define tasks and to capture end user interactions, needed to be extensively rewritten in order to deal with touch events. Moreover, it was also necessary to extend the code in order to process HTML5 tags correctly.

Another issue was related to the fact that mobile browsers need to present a web application even when it is not designed to be used in a mobile device. For this reason, for each touch event, browsers also trigger equivalent mouse events. For instance, when the user touches the screen, besides triggering tap events, the browser also triggers click events, which allow the application to function even if it does not support touch events. Therefore, it was necessary to filter the mouse events before sending the data to the server.

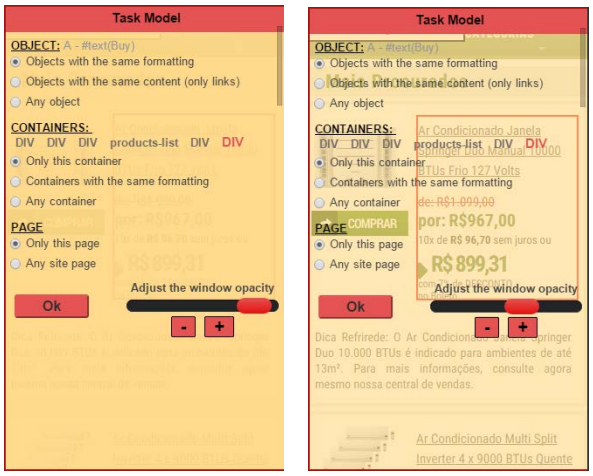
3.2 Task Definition

As was true for desktop web applications, mobile applications also benefit from the COP interface model in order to save work when defining tasks. By exploiting the features of our interface model, a single task may be used to represent several similar tasks. In order to achieve this goal, evaluators need to identify which events need to be generalized.

In order to activate the generalization features in USABILICS, we implemented a mechanism based on the usage of the ctrl key. When the evaluator wants to generalize an event — a click, for instance — all she needs to do is to press the ctrl key before performing the click. In this way, she is prompted for the generalization options after the click. This same procedure is, of course, not feasible in mobile devices.

In order to provide the same functionality for the mobile environment, we extended the UsaTasker tool, making it a tool that can be used to define tasks both in the desktop and in the mobile environments. The new version of UsaTasker is able to detect a mobile web application and, in this case, it blocks the screen whenever an event is performed during the capture of a task. The blocking was implemented by

displaying a translucent panel on top of the current window. As shown in Figure 1, this panel presents the generalization options, allowing to generalize the event to objects with same formatting and/or content. Moreover, it is also possible to generalize the event to other objects within the same container or in other containers. To make it easier for identifying the existing containers, the application highlights the borders of the container when it is selected on the panel. The option for the translucent panel was made for making it possible to visualize the selected containers in the application's interface. As can be noticed on Figure 1, buttons "+" and "-" are provided in order to allow users to increase and decrease the opacity of the panel.



(a) Normal opacity (b) Lower opacity

Figure 1: UsaTasker displaying the options of the COP model.

By finishing the configuration of the captured event, the evaluator needs to touch the "ok" button in order to close the translucent panel. After closing the panel, the Javascript application triggers the intercepted event. For instance, if the event is touching a link, the target page will be loaded in the screen after the translucent panel is closed. During the capture of tasks, a small panel on the upper left side of the screen is displayed. This panel presents a "Close" button that must be pressed when the capture of the task finishes.

When the evaluator finishes the task definition, UsaTasker presents the captured events graphically, as shown in Figure 2. This visual feedback is important because it provides a way for verifying if each event that composes a task was correctly recorded. In Figure 2, each box represents an event, and the red directed edges indicate the order of each event within the task. Besides viewing the details of each event, the evaluator may delete an event, if she considers that this event is irrelevant in the optimal path of the task. To perform the deletion of an event, all the evaluator needs to do is to click on the x icon on the top of the desired box.

Besides providing facilities for the visualization of captured tasks, UsaTasker also allows the management of tasks, allowing (i) the definition of sequence of events in which each event may occur in any order; (ii) the definition of sequence

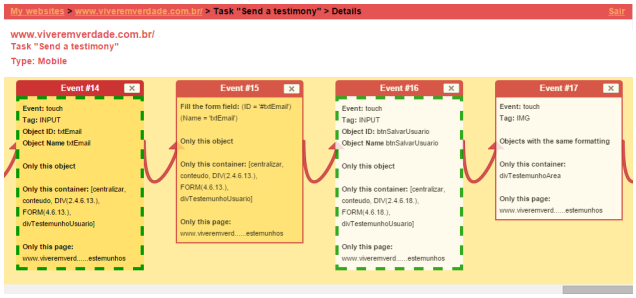


Figure 2: Captured events in MOBILICS

of events that may be repeated several times; and (iii) the definition of optional events. In this way, it is possible to specify, for instance, that the fields of a form may be filled in any sequence (i), that certain steps of tasks, such as putting products on a shopping cart may be repeated several times (ii) and, finally, that a certain field of a form is optional and, therefore, may be left blank (iii). UsaTasker's management tools are detailed in [16].

3.3 Logging

Logging in MOBILICS is quite similar to that in USABILICS. The main difference is related to the fact that mobile browsers may trigger both mouse and touch events when a touch is performed. Therefore, in order to reduce the volume of data that needs to be logged, the Javascript client application filters out the mouse events and only logs touch events. As in USABILICS, logs are compressed and sent to the server.

3.4 Task Analysis

In Section 3.1 we pointed out that touch events are reported differently among browsers and mobile platforms. We also mentioned that, as mobile browsers need to present both mobile-enabled websites and desktop-based websites, they trigger both mouse and touch events. In addition, older browsers report only mouse events, even when the web application is mobile-enabled.

To deal with this problem, we had to define a set of similar events. Therefore, if the next event in the optimal path of the task is a *tap* and we find a *click* in the log, we must consider this event as a correct one, because *tap* and *click* are similar events. *Drag* and *scroll* are also similar events, since mobile-enabled browsers triggers both drag and scroll while older browsers only trigger *scroll* events. Finally, *scroll* and *swipeup/down* are also similar for the same reason.

In order to explain how similar events are processed, we need to recall the computation of the usability index, detailed in [14]. When comparing an event performed by the user and the corresponding event defined by the evaluator, USABILICS produces a similarity value between 0 and 1. Our approach to calculate this value is based on the importance of each COP model concept associated to the event. Therefore, we apply weights for these concepts: 0.1 for the page, 0.3 for the container and 0.5 for the object. We also apply 0.1 for the correctness in the type of the event. Therefore, if the event is performed in the correct page, we compute

0.1, if it is performed in the right container, we sum 0.3, if it is performed in the right object, we sum 0.5, and if the type of event is correct, we sum 0.1. Thus, when an event is correctly accomplished, the result is 1.

When the value of the comparison of an event is 0.9 and the event was performed in the right page, everything is correct, except for the event. In this case, we need to check if the event is similar. If so, we correct the value of the comparison to 1.0 as, in fact, the event is the same.

3.5 Recommendations

Early experiments with iPads performed by Nielsen anticipated some of the problems of today's mobile web applications [11]. In general, the interaction design varies across mobile web applications, making it difficult for users to transfer their skills from one application to the other. As a result, users tend to have an exploratory behavior, especially when using an application for the first time.

Even experienced mobile device users face difficulties using mobile web applications. Touching a target, for instance, may be very challenging for users with large hands using small form factor devices. Even finding a target may be hard in these type of devices. MOBILICS can detect problems in finding or interacting with a target by analyzing the events before the interaction. Unexpected zooming or scrolling before a touch event may indicate difficulties with finding and/or touching a target.

We also analyze the finger footprint in order to detect problems. If the user is touching a container — which is obviously not a target object — it may be possible that s/he is experiencing difficulties touching a target. In this case, the target (a link or button, for instance) needs to be enlarged.

Table 1 presents detected usability issues and the recommendations provided by MOBILICS in order to fix the problems. For each problem, we also present the expected event and the actual performed event (wrong action).

The recommendation procedure in MOBILICS was redesigned in order to be extensible, making it possible to add new recommendations by defining a *problem* and its *solution*. A *problem* is an action that does not belong to the optimal path of a task, as the ones shown in the second column of Table 1. A *solution* is a recommendation that fixes the problem. Examples of solutions are shown in the third column of the table.

After analyzing a task and presenting its usability index, MOBILICS provides a report containing the recommendations, when fixes are needed. Besides the recommendations shown in Table 1, the report also presents links to content available on the web addressing good programming practices that improves the usability of web and mobile applications.

4. EVALUATION

A common procedure for assessing the effectiveness of usability evaluation tools is recruiting users to perform specific tasks. This approach has some drawbacks. First of all, as recruited users are normally graduate or undergraduate students, they have a better education background than the general public. Moreover, these students are familiar with web and mobile applications, which is not the case for the average user. Lastly, they receive specific instructions about the task they have to perform, which does not happen with the real user, who has to browse the website to discover how to perform a task. Therefore, we think that studies that rely on recruited users may provide unreliable results.

Instead of recruiting users, we decided to evaluate MOBILICS using a popular website called *Living in Truth — Viver em Verdade*, in Portuguese (www.viveremverdade.com.br) — which is a biblical studies website that has more than 1,700 visits per day. This website has been designed for desktop access, but we were aware that, as happens to many websites as of today, a number of users visit the site using a mobile device.

In order to evaluate if MOBILICS could help adapting *Living in Truth* to be a mobile friendly web application, we first chose a task available on the site. The chosen task is called *Testimony*, in which the user publishes a testimony about his/her faith. We chose this task because although it is easy to perform, it requires several different user interactions across 5 different pages. Therefore, we can evaluate if all 5 pages need to be fully adapted or if, as we advocate, less effort is necessary to adapt the task.

For this experiment, we used MOBILICS to collect user logs over 14 days. In order to avoid privacy problems, the website displayed a message in the first page warning the users about the fact that their interactions were being captured "to improve the quality of the website". Except for this warning message, the website was exactly the same as before.

The logs showed 265 attempts to execute the task — i.e. the task was initiated 265 times. However, only 66 of these 265 attempts were fully completed. As the website is open to the public, it is expected that people explore the site at will, so the number of interrupted tasks is normal. Of the 66 completed tasks, 41 were performed using a PC and 25 using a mobile device.

For the 41 tasks performed using a PC, MOBILICS computed an usability index of 0.8, which suggests that the site has good usability for this task in the desktop environment. Results above 0.75 and below 0.9 indicate that while there is room for improvement, usability is good overall [15]. On the other hand, when MOBILICS analyzed the 25 tasks performed using mobile devices, the usability index was only 0.5. This result was expected, as the tested website was designed for desktop use.

When MOBILICS detects usability issues, it provides recommendations in order to fix them. To explain the rationale behind each recommendation, it is worth to understand the steps of the task being evaluated. To publish a testimony, the user selects the "Testimony" button on any page of the website. Next, she is asked to provide her state and city using combo boxes, and her name and email using text boxes. Then, the user is asked to select the type of testimony. Cure, conversion, and family are some of the options. Finally, a text area is displayed so that she can fill in her testimony.

Next, we present the events of the optimal path for this task. It is worth noticing that an event may be explicitly triggered

Table 1: MOBILICS recommendations

(E)xpected (P)erformed	**Detected action**	**Recommendation**
E: Tap on component X. P: Tap on component Y.	**D1**: User is tapping a wrong component.	**R1**: Increase the space between components. Highlight component X in the page.
E: Tap on component X. P: Tap within the container of component X.	**D2**: User is tapping the container area.	**R2**: Increase the component and/or the tapping area associated to the component.
E: Tap on component X. P: Pinch or double tap within the container of component X.	**D3**: User is magnifying the region where a component is located.	**R3**: Increase the tapping area of the component, the size of the component and/or the space between this component and others.
E: Tap on component X. P: Drag within the container of component X.	**D4**: User is searching for a component.	**R4**: Place the component in a visible area of the page.
E: Tap on an input field. P: Drag within a container that contains an input field.	**D5**: User is searching for an input field.	**R5**: Place the input field in a visible area of the page. Use visible labels, increase the size of labels and fields.
E: Page load in normal time. P: Page load slow.	**D6**: Page is taking too much time to load.	**R6**: Supress figures or use media queries to load images with less resolution. Compress Javascript and CSS files. Remove plugins.
P: Horizontal dragging.	**D7**: Horizonal scrolling.	**R7**: Adjust the layout of the page in order to use only the available viewport. Use responsive layout in order to automatically adapt the page to the size of the screen.
P: Vertical dragging.	**D8**: Vertical scrolling.	**R8**: Diminish and condense the contents of the page or break the page in two.
P: Pinch after a page load.	**D9**: User needs to magnify the screen as soon as it loads, probably because she is not able to read the contents of the page.	**R9**: Change viewport properties so that the page scales correctly.

by the user – by clicking or touching an interactive element – or triggered by the system as a response to an user event (open/load a page, for instance). The optimal steps are:

1. Open page (any page of the website);
2. Load page;
3. Click/tap the “Testimony” button of the menu;
4. Close page;
5. Open page “Testimony”;
6. Load page “Testimony”;
7. Click/tap the “State” combo box;
8. Select a state in the combo box;
9. Click/tap the “City” combo box;
10. Select a city in the combo box;
11. Click/tap the “Next” button;
12. Click/tap the text field “Name”;
13. Fill in the text field “Name”;
14. Click/tap the text field “email”;
15. Fill in the text field “email”;
16. Click/tap the “Next” button;
17. Click/tap one of the buttons that represent the type of testimony;
18. Click/tap the text area for the testimony;
19. Fill in the text area;
20. Click/tap the button “Submit testimony”;
21. Form is closed (data is sent).

By comparing the sequence of each recorded task to the optimal path of the task, MOBILICS is able to detect the wrong actions performed by the end users. When the website first loads, the user is supposed to click or tap the ”Testimony” button (event 3). If this event is not present in the sequence of events of the end user, it might be a problem. So we have to analyze the logged events *before* the click/tap event in order to find wrong actions. In the case of our example, the logs report that most users performed dragging (zoom) and horizontal scrolling. This is because the page was formatted for desktop access. When the user opens the page using a mobile device, the page is ”squeezed” to fit in the viewing area, making each component too small to see or to tap. Therefore, the user needs first to drag (zoom) in order to see the component she is looking for (detected action D3 in Table 1). But, as the buttons get bigger, they do not fit in the viewing area, so users need to use horizontal scrolling in

order to view all the buttons (D7 in Table 1). Considering the detected wrong events for page 1, MOBILICS recommendations for fixing the problems are R3 and R7.

The layout of the second and third pages are almost the same. Both of them collect information in two combo boxes. MOBILICS reports the wrong actions D2, D3 and/or D7 before the events 7 and 12 for at least half of the recorded interactions. Bearing in mind that these pages were designed for the desktop, and considering that the combo boxes are located in the middle of the page, when the user opens the page in a small screen device, she will be able to see the first combo box. However, it appears fairly small, so she can miss the tap (D2). After that, she may decide to pinch (zoom) for enlarging the component (D3). If the zoom is excessive, the component may stay outside of the viewport, therefore it might be necessary to do some horizontal scrolling (D7). Of course, these three wrong actions do not appear in this order in every interaction that presented problems. In some cases, the user missed the tap several times, until she was able to do it, so D3 and D7 do not appear in the log. In other cases, the user noticed from the begining that the target is too small, so she starts zooming the page. In this case, we see no D2 in the log. To solve the detected usability problems in pages 2 and 3, recommendations are R2, R3 and R7.

When posting a testimony, the user is asked to select its type. The website presents the types in six large buttons in page 4. These buttons fit well in a desktop screen. However, when using a small screen device, users needed to make both horizontal (D7) and vertical (D8) scrolling before making their choice (event 17 of the task). Therefore, the recommendations for fixing the problem are R7 and R8. The user logs for the last page (page 5) did not show usability problems. Therefore, this page may be used as is in the mobile verson of the website.

The detected wrong actions point out that the usability problems are not widespread. Some pages present several usability issues, while other present few or even no issues. Therefore, this experiment backs up our argument that a website designed for the desktop does not need to be reconstructed from scratch in order to present a good usability in mobile devices.

As MOBILICS is able to identify an usability issue and the component or components associated to this issue, it is very straight forward for the developer to fix the pointed problems. So, following the provided recommendations, we built a new version of the website and, once again, performed the same test. At this time, we decided to wait until 25 completed tasks were made using mobile devices. After achieving this total, we performed again the log analysis. As expected, the index has risen significantly, reaching 0.75. This value is very close to the usability index computed for the desktop version of the website, which was 0.8. Therefore, we claim that our adaptation approach based on the recommendations provided by MOBILICS was successful.

5. RELATED WORK

As discussed in the Introduction, existing approaches for automatically adapting desktop-based websites for mobile devices are limited to specific adaptation scenarios. These techniques usually change the navigational model of the existing applications [2, 6, 7], or change its contents [13, 1]. Therefore, the resulting mobile website is significantly different from its desktop version, both in terms of appearance and functionality.

Another work that is based on changing the contents of the website is PageTailor [3]. However, instead of automatically trying to define irrelevant elements, this approach lets end users adapt the layout of Web pages by removing, resizing and moving page elements. PageTailor records the user customizations and automatically reapplies them on subsequent visits to the same page, or to similar pages, on the same website. A problem of this approach is that it requires significant effort from the end user in order to adapt a web page. The customization process takes about 10 minutes per page, which may be too much for a website containing dozens of pages. Moreover, page changes can cause customizations to break.

Following PageTailor's concept that the user may guide the adaption process, Highlight [10] allows users to demonstrate interactions with a web page that are then used to create a version of the original page adapted to mobile devices. This approach exploits a remote control metaphor in which the mobile device controls a fully functional browser that is embedded within a proxy server. The main drawback of this approach is that it does not allow users to customize elements themselves, only the interactions with them. Therefore, the customization process is limited.

Another tool that exploits the end user customization of web pages is Chickenfoot [4]. The goal of this work is to create an opportunity for end-users who want to automate and customize their web experiences. This tool aims at supporting end users to script common web interactions within and across web sites based on the visual appearance of DOM elements using keyboard pattern matching. In spite of being an interesting approach in order to allow web page customizations, this work does not offer any support for adapting interfaces to mobile devices.

Among all related work, W3Touch [9] is specially relevant because it exploits usability evaluation techniques to automate the desktop-to-mobile migration of websites. W3Touch instruments websites using Javascript applications in order to perform logging of the end users interactions. The collected information is then used to assess usability metrics, such as the amount of zooming events and missed clicks. Based on these metrics and on configurable thresholds, adaptation rules may be written in order to solve detected usability problems.

W3Touch logging approach is very similar to what we use in MOBILICS. However, our usability metric is task-oriented, while W3Touch use an event-oriented approach for defining metrics. We advocate that our approach is more efficient in order to assure the evaluation of the functional requirements of a web application. W3Touch approach only checks if an atomic event, such as touching a link, is happening as expected. On the other hand, our approach is able to verify if a whole task is being executed as expected. By evaluating all the existing tasks of a website, we are able to say that all functional requirements of the web applications were

exercised, which is an important feature for any application evaluation toolkit.

Finally, it is worth to compare our work to responsive design tools such as Boostrap and Webflow. There is no doubt that responsive design is the way to go in order to make adaptable websites. However, in order to use these tools, the developer needs to build new applications from scratch. Our approach, instead, allows the developer to benefit from an existing desktop-based web application in order to build its mobile-enabled version, thus saving time and development effort.

6. CONCLUSION

This paper presents a new approach for supporting the adaptation of web applications to the mobile environment. Our approach builds on the hypothesis that significant portions of code from a desktop-based web application can be used to create a mobile-enabled version of this application, thus, saving time and effort from building a mobile version from scratch.

The challenge of reusing content from desktop-based web applications is determining which parts can be reused and which parts need fixes. Moreover, support for making the needed fixes is crucial, otherwise the effort for fixing issues may be as time consuming as building a new mobile-enabled website.

In order to provide a cost-effective solution for supporting the desktop-to-mobile migration of web applications, we developed MOBILICS, a system that supports the usability evaluation of mobile websites. MOBILICS' task-based evaluation procedure allows the detection of usability problems in the execution of tasks and points out the interface elements that presents usability issues. Based on this procedure, the system provides detailed recommendation on how to fix the pointed usability problems.

We used our approach for creating a mobile-enabled version of an existing desktop-based web application. This experiment allowed us to confirm our hypothesis that significant amount of code can be reused in the adapted mobile web application. The experiment also demonstrated that our recommendations were effective in order to support the adaption of the desktop-based application to the mobile environment.

7. ACKNOWLEDGMENTS

The authors would like to thank the Brazilian National Council for Scientific and Technological Development (CNPq) for the financial support.

8. REFERENCES

[1] P. Baudisch, X. Xie, C. Wang, and W.-Y. Ma. Collapse-to-zoom: Viewing web pages on small screen devices by interactively removing irrelevant content. In *Proc. of the 17th ACM Symp. on User Interface Software and Technology*, pages 91–94. ACM, 2004.

[2] T. W. Bickmore and B. N. Schilit. Digestor: Device-independent access to the world wide web. In *Selected Papers from the Sixth International Conference on World Wide Web*, pages 1075–1082. Elsevier Science Publishers Ltd., 1997.

[3] N. Bila, T. Ronda, I. Mohomed, K. N. Truong, and E. de Lara. Pagetailor: Reusable end-user customization for the mobile web. In *Proc. of the 5th International Conference on Mobile Systems, Applications and Services*, pages 16–29. ACM, 2007.

[4] M. Bolin, M. Webber, P. Rha, T. Wilson, and R. C. Miller. Automation and customization of rendered web pages. In *Proc. of the 18th Annual ACM Symposium on User Interface Software and Technology*, pages 163–172. ACM, 2005.

[5] D. Bosomworth. Mobile marketing statistics 2015. http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics.

[6] O. Buyukkokten, H. Garcia-Molina, A. Paepcke, and T. Winograd. Power browser: Efficient web browsing for pdas. In *Proc. of the SIGCHI Conf. on Human Factors in Comp. Systems*, pages 430–437. ACM, 2000.

[7] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proc. of the 12th Intern. Conf. on World Wide Web*, pages 225–233. ACM, 2003.

[8] R. Hof. Google research: No mobile site = lost customers. http://www.forbes.com/sites/roberthof/2012/09/25/google-research-no-mobile-site-lost-customers.

[9] M. Nebeling, M. Speicher, and M. Norrie. W3touch: Metrics-based web page adaptation for touch. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2311–2320. ACM, 2013.

[10] J. Nichols, Z. Hua, and J. Barton. Highlight: A system for creating and deploying mobile web applications. In *Proc. of the 21st Annual ACM Symposium on User Interface Software and Technology*, pages 249–258. ACM, 2008.

[11] J. Nielsen. ipad usability: First findings from user testing. http://www.nngroup.com/articles/ipad-usability-first-findings.

[12] J. Nielsen. Mobile site vs. full site. http://www.nngroup.com/articles/mobile-site-vs-full-site.

[13] V. Roto, A. Popescu, A. Koivisto, and E. Vartiainen. Minimap: A web page visualization method for mobile phones. In *Proc. SIGCHI Conf. on Human Factors in Comput. Systems*, pages 35–44. ACM, 2006.

[14] L. G. Vasconcelos and L. A. Baldochi. USABILICS: remote usability evaluation and metrics based on task analysis (in portuguese). In *Proc. of the 10th Brazilian Symposium on Human Factors in Computing Systems*, pages 303–312. Brazilian Computer Society, 2011.

[15] L. G. Vasconcelos and L. A. Baldochi. Towards an automatic evaluation of web applications. In *Proc. of the 27th Annual ACM Symposium on Applied Computing*, pages 709–716. ACM, 2012.

[16] L. G. Vasconcelos and L. A. Baldochi. Usatasker: A task definition tool for supporting the usability evaluation of web applications. In *Proc. of the IADIS Internat. Conf. WWW/Internet*, pages 307–314, 2012.