Towards an automatic evaluation of Web applications

Leandro Guarino de Vasconcelos
Universidade Federal de Itajubá
Cx. Postal 50 - 37500-903 - Itajubá-MG
le.guarino@gmail.com

Laércio Augusto Baldochi Jr.
Universidade Federal de Itajubá
Cx. Postal 50 - 37500-903 - Itajubá-MG
baldochi@unifei.edu.br

ABSTRACT

Evaluating the usability of computer applications using traditional laboratory-based tests is costly and time consuming. For modern Web applications, usually developed, tested and deployed in "Internet Time", this approach is simply not feasible. A more effective way to evaluate the usability of Web applications consists in gathering information from the user's interactions and automatically processing this data in order to detect usability problems in the execution of predefined tasks. The reported solutions based on this approach usually fail on providing efficient tools for the definition of tasks, specially in large and dynamic Web applications. In order to tackle this problem, we developed USABILICS, a system targeted for the automatic remote evaluation of usability based on an interface model. The proposed model allows the definition of tasks using a simple and intuitive approach, which can be applied to large and dynamic Web applications. USABILICS analyzes the execution of tasks by calculating the similarity among sequence of events produced by users and those previously captured by evaluators. Based on this analysis, USABILICS provides an usability index, as well as recommendations for solving usability problems detected on the execution of each task.

Categories and Subject Descriptors

H.5.2 [**User interfaces**]: Evaluation/methodology

General Terms

Human Factors, Design, Measurement

Keywords

Remote usability evaluation, Interface model, Task analysis, Usability problems, Usability index

1. INTRODUCTION

The Web has been experiencing a continuous and impressive growth in the last decade. One of the reasons for this growth is the availability of tools and facilities for the creation and publishing of information on the Web. More than simply facilitating the creation of home pages, these tools makes it straightforward to build full-fledged Web applications. As a result, more and more people without a Web development background are publishing content on the Web. In this scenario, usability principles are rarely considered in the development process, resulting in Web application interfaces that tend to present usability problems.

For modern Web applications, usually developed, tested and deployed in "Internet Time", regular approaches for the evaluation of usability, based on laboratory tests, tend not to be appropriate, as they demand an amount of effort and time that developers are not willing to spend. An option to tackle this problem is using remote automatic or semi-automatic usability evaluation tools. According to Andreasen et al. [1], these tools allows evaluating a large number of users by a low cost, as users and evaluators may be separated in time and space. Paternò & Paganelli [8] showed that automatic evaluation provides useful information for the identification of problems in Web interfaces.

Tools developed for providing remote usability evaluation usually capture log information on the client using additional applications that run in background, gathering information about the user's interaction [6]. The captured logs are sent to server-side applications, where they may be processed in different ways. An approach that is effective towards identifying usability problems consists in analyzing the captured events according to a task model, which is previously defined for the application under evaluation. The comparison between the sequence of events performed by the end user and the sequence of events defined on the model may indicate eventual usability problems. WebRemUSINE [7] and AWUSA [12] are examples of tools that exploits this approach. These tools, however, use procedures for defining tasks that do not scale well for large and dynamic Web applications, in which tens or even hundreds of tasks need to be defined.

Towards addressing this problem, we developed USABILICS [16], a system that aims at facilitating the remote evaluation of usability in large and dynamic Web applications. In order to address the drawbacks of the task models proposed in the literature, USABILICS provides an interface model that supports the definition of tasks in a simple and intuitive way. Based on this model, evaluators are able to define tasks by simply interacting with the application's graphical interface, in the same way end users are supposed to do. Another important aspect of our model is the fact

SAC'12 March 25-29, 2012, Riva del Garda, Italy.

that it considers the similarity among the possible paths of a given task, allowing a generic approach for the definition of similar paths, what reduces considerably the time taken to define tasks, as a group of similar tasks may be represented by a single generic task.

The evaluation of usability is performed by comparing the sequence of events captured during the definition of tasks to the corresponding sequence gathered from the user interactions. In order to capture these interactions, USABILICS provides a tool that executes in the client browser performing the logging of events. Logs are periodically compressed and sent to a server-side application which executes an algorithm that uncompress the logs and, based on our interface model, searches the logs for sequences of events that may correspond to attempts for accomplishing a pre-defined task. Whenever a task is identified, its sequence of events is compared to the sequence recorded for the definition of the corresponding task. As a result, we provide a metric, which we called *usability index*, that allows evaluating both the efficiency and the effectiveness of each task of a Web application and, as a consequence, allows evaluating the Web application in its entirety.

USABILICS has been tested in different Web applications. The obtained results, when compared to laboratory-based tests, shows that our approach is effective towards identifying tasks that present usability problems, and, most importantly, the extent of these problems. However, simply pointing out a problem may not be enough when the application developer is not experienced in usability evaluation. To address this problem, we extended USABILICS in order to provide automatic recommendations that aim at improving the usability of applications that present low usability indexes. Experiments performed with the new version of the system, report that these recommendations were effective towards raising the usability index of tested applications.

This paper is organized as follows. Section 2 presents USABILICS in detail, explaining its interface model, the approaches for task definition and task analysis, and, finally, presenting how the system compute the usability index for tasks and applications. Section 3 presents our approach for implementing automatic recommendations to tackle usability problems in Web applications. Section 4 presents a summary of the literature in the field of remote usability evaluation, comparing and contrasting USABILICS to other relevant systems. Lastly, Section 5 brings our concluding remarks and discuss future work.

2. THE USABILICS SYSTEM

Our approach for the usability evaluation of Web applications is based on a three-step procedure: (i) task definition; (ii) gathering of data produced by users while performing the defined task; and (iii) task analysis. The definition and the analysis of tasks are complex issues, specially in large Web applications. To cope with this problem, we developed an interface model which abstracts structural patterns we found on Web Applications. Our model, called COP, is presented on next section. Following, we detail the steps of our usability evaluation approach.

2.1 The COP interface model

A Web application is composed of a collection of pages, which in turn are composed by elements such as hyperlinks, tables, forms, etc. These elements, specially in dynamic Web applications, are commonly shared among two or more pages. This pattern is intensified when *Content Management Systems (CMS)* and templates are used. Our approach to provide usability evaluation for Web interfaces exploits this pattern in order to facilitate both the definition and the analysis of tasks.

According to the 4.01 HTML specification, components of a page may be grouped by the elements DIV and SPAN. Moreover, the specification defines that the attributes *id* and *class* are identifiers for page elements, and the *id* attribute needs to be unique within a page. Considering both these properties and the pattern we observed in the structure of Web applications, we proposed the COP model, an interface model based on three main concepts: ***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. Examples of containers are table cells, layout divisions in a page, lists and forms. Finally, a *page* is an interface that contains one or more containers.

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. Finally, it is worth noticing that objects and containers may appear in one or more pages of a Web application. Table 1 abstracts the details of the COP model. The left column of the table presents the three main concepts of the model and the right column depicts the options that are applicable to each concept when defining a task.

Table 1: Concepts of the COP model

Concept	Options
Object	A single object
	Objects with the same content
	Objects with the same formatting
	Any objects
Container	A single container
	Containers with the same formatting
	Any containers
Page	A single page
	Any pages

The structure of the COP model is presented in Figure 1. Lines that connect the graphical elements represent a possibility of association among them. A *Unique object* is an object that is identified in a unique way. Unique objects, as well as similar objects in terms of formatting or content may be kept within a single container, which is also identified in a unique way (*Unique container*), or in two or more *Similar container*s. Figure 1 also shows that a container may belong to a unique page or take part in several pages of a Web application.

Considering the W3C Recommendation for the construction of hypertext-based documents [9], it is possible to say that the concepts of the COP model are sufficient to identify any element of a Web interface. Therefore, we advocate that our model is adequate for the definition of tasks in Web applications. Defining a task means specifying an optimal path for accomplishing the task. An optimal path for a given task

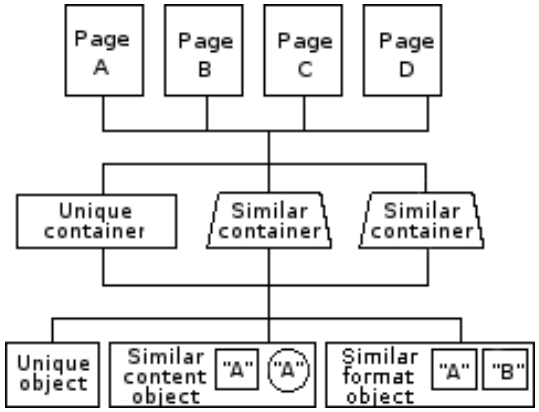

Figure 1: The COP model

is the sequence that presents the smaller number of events to perform the task. Exploiting this model, USABILICS provide a simple and very intuitive way to define a task: it simply captures the interaction of the developer while she performs the task.

Defining tasks in such a way that they can be evaluated in an automatic or semi-automatic fashion is challenging because a task may be accomplished using different paths. Therefore, the definition of tasks in large applications may be exhaustive and the problem tends to be aggravated in dynamic Web applications, where a lot of elements are usually shared among different pages. As an example, consider the definition of the task *Buying a product* in an e-commerce Web site that has 10,000 products for sale. Individually specifying all the possible paths to perform this task is virtually impossible. The COP model was tailored to deal with problems like this. Our approach allows the automatic definition of alternative paths as it considers the similarities among objets and containers, resulting in a significant reduction of time and effort for the definition of tasks, specially in large and dynamic Web applications. For this reason, the USABILICS system is able to define a single optimal path for the task *Buying a product* that will work fine for buying any of the 10,000 products of the application we just presented.

2.2 Task definition

Defining tasks is the starting point for evaluating the usability of a Web application in our system. USABILICS provides a tool which aims at making the definition of tasks easier. Similar to Google Analytics [4] and WELFIT [11], this tool is a server-side application that allows users to subscribe themselves as evaluators of specific Web applications. For each subscribed application, USABILICS provides a Javascript code which needs to be inserted in the pages targeted to be evaluated.

In order to define the optimal path for a task, the evaluator logs in the server-side application and fills in the name and a description for the task she wishes to define. Following, the selected application is loaded in a new window, making it possible to start recording the task. Upon finishing the task, the evaluator closes this window, stopping the recording process. Therefore, the evaluator defines a task by simply using the application, in the same way end users are supposed to do. During the task definition, when the evaluator performs an action, such as clicking a button, she is prompted with options that allows the specialization or the generalization of the event, as follows:

- consider only this object;
- consider objects with same content;
- consider objects with same formatting;
- consider ANY objects.

For containers and pages, similar options are prompted to the evaluator, as depicted in Table 1. Therefore, if we consider as an example the event *Putting a product in the shopping cart*, in an e-commerce application in which all displayed products shows a standardized *Buy* button, when the evaluator clicks this button and is prompted for her choices, she may chose *Consider objects with same content*, *Consider containers with same formatting* and *Consider ANY pages*. Using these generalization options, the defined task will be applicable to any product of the e-commerce application.

For a detailed explanation regarding the definition of tasks, refer to [16].

2.3 Automatic data gathering

Similarly to WELFIT [11] and WAUTT [10], USABILICS provides a Javascript application that runs attached to the pages of the applications that are targeted to be evaluated. This application captures mouse movements, scrollbar movements, resizing of windows and several other events related to the interaction with components of application's interface, as well as events generated by the pages of the application, such as *load* and *unload*. However, differently from WELFIT and WAUT, we also capture CSS attributes from the elements of the interface with which the user interacts, as the positioning, the colors, the fonts and other features related to the formatting of elements directly interfere in the way users interact with these elements. Moreover, we use CSS attributes together with the COP model in the analysis of tasks, when comparing objects and containers that present similar formatting, allowing to treat them generically. We also capture a timestamp associated to each event in order to check for interruptions during the execution of tasks.

More recently, with the increasing popularity of technologies such as AJAX, there is a trend for the development of rich Web interfaces, which allows loading content asynchronously. In order to address both regular and rich interfaces, our approach supports a fine granularity for the gathering of data, aiming at making it easier to find usability problems. Therefore, USABILICS gathers, for instance, the dimensions of the window of the Web browser for each captured event, as the user may resize the window causing a modification in the placement of elements, what may hide a given element and, as a result, delay the accomplishment of a task.

In order to identify problems associated to the usage of specific browsers, including problems associated to the usage of plugins, USABILICS gathers detailed information regarding the browser and its plugins. The information about plugins are important, since the absence of a needed plugin may prevent a video or an animation to be displayed, affecting the behavior of the user while she tries to accomplish a task.

USABILICS' client side tool was optimized to consume low CPU resources and to have a small footprint on memory. This measure aimed at making the tool lightweight, so that it remains almost imperceptible for the user while she performs a task. The collected data is compressed and sent to a server application, which pre-process the information and stores it in a database.

2.4 Task analysis

According to ISO 9241-11 [5], usability refers to the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. This definition is related to five concepts: (1) user: person that interacts with the system; (2) context of use: users, tasks, equipments (hardware, software, materials), physical and social environment in which the system is used; (3) effectiveness: entails the accuracy and the completeness with which users achieve specified goals; (4) efficiency: the relation between effectiveness and the amount of resources needed to accomplish the goals; and (5) satisfaction: measures the freedom from discomfort, and positive attitudes towards the use of the product. Our approach addresses concepts 2, 3 and 4. USABILICS performs task analysis comparing the sequence of events recorded for a given task and the correspondent sequence captured from the end user interactions. The similarity between these sequences provides a metric of efficiency. The percentage of completeness of a task provides a metric of effectiveness. The context of use is inferred based on the gathered data about the operational system, the browser and the speed of the connection.

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.

For the definition of a task, events such as *mouseover*, *mouseout*, *mousemove*, among others, are not considered, as they do not correspond to decisions of the end user. However, these same events need to be captured during the end user interaction, as they are important to detect usability problems. Therefore, considering that (i) the set of the collected information that needs to be compared is slightly different, and (ii) the comparison process takes into account the generalization options of the COP model, we used four main factors to calculate the similarity between two events: the event type, the object in which the event was targeted, the container of this object, and the page in which the event occurred. Those factors are important, for instance, to compare an event *mouseover* related to the end user interaction and a *click* event found in the task definition. Besides not being identical, these events are similar, as positioning the mouse over an object may indicate that the user was considering clicking on the object. This example shows that providing a binary answer (true or false) for a comparison between two events is not appropriate.

Therefore, when comparing an event performed by the end user and the corresponding event defined by the evaluator, we produce 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 as follows: 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 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 – a click, for example – we sum 0.1. In this case, the result is 1 – the event was correctly accomplished. It is worth noticing that weights related to containers and objects have larger values because it is important to make a clear distinction between events performed close to target objects (considered similar to the defined events) and those performed in other containers or in objects placed far from the target (considered non-similar to defined events).

USABILICS checks the sequence of events for the task and, for each event considered similar, increments the completeness counter. In this way, it is possible to determine whether a task has been successfully completed, and, otherwise, calculate the percentage of completeness. Generalizing the similarity between events, we get a measure of similarity between the sequence of events defined in the task and the sequence of events produced by the end user. This similarity of sequences is given by the following formula:

$$sim(Si, Sj) = (SS/QE) * PC$$

where, Si is the sequence of events resulting from the end user interaction; Sj is the sequence of events defined for the task; SS is the sum of the similarity values of events in Si and Sj; QE is the quantity of events in Si; and PC is the percentage of completeness of the task.

By exploiting the similarity among sequences, we proposed an usability index, which is given by the following formula:

$$UsabilityIndex = (\sum sim(Si, Sj))/QI$$

where QI is the number of times task I was performed by end users.

The proposed index is an important tool for Web developers, as it indicates in a clear and summarized way the usability of a task. The redesign of an interface, for example, may benefit from this index, as developers may compare the index of tasks before and after the redesign, ensuring that updates do not harm the usability of the application.

In order to validate our approach, we tested USABILICS in Web applications from different domains, such as e-commerce, e-learning and personal blogs [16]. In total, seven different applications have been used in our experiment. After defining tasks, we collected the corresponding data from these sites for more than six months. We then chose specific tasks – those that presented a larger number of subsequences and, therefore, may be considered more difficult to be performed – in order to compare the index computed for these tasks and the results of laboratory-based usability tests.

The tests were performed using 14 participants, from different ages (19 to 42 years-old), different professions and having different computer skills. It was found compatibility between the results of the tests and the corresponding index for 100% of the compared tasks. Following, we present the results observed for two different tasks.

For the task *Correcting an assignment* in an e-learning application, USABILICS computed an index of 0.1348 (13.5%). Analyzing the laboratory-based tests, it was found that 60% of the users were not able to perform the task using the optimal path and 40% of the users were not able to accomplish the task at all. Therefore, the low usability index is correct – this task actually presents serious usability problems. In the other hand, the task *Commenting a post* in a blog application presented an usability index of 0.7993 (79.9%). The outcome of the laboratory-based tests showed that 100% of the end users were able to perform the task using the optimal path, however it was found wrong actions related to mouse movements and scrolling, what confirms the results of the calculated index.

3. AUTOMATIC RECOMMENDATIONS

The usability index is a valuable information for Web developers, as it points out the tasks that present usability problems, and, most importantly, the extent of these problems. However, for developers who are not experienced in usability evaluation, pointing out a problem may not be enough, as these developers may not be able to perform the improvements required for solving the problem. In order to address the needs of these developers, we extended USABILICS so that it suggests measures to tackle the pointed problems.

As pointed out by Vermeeren et al. [17], when comparing the sequences of events that compose a task it is possible to find out three different situations which indicate the occurrence of wrong 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. Therefore, we extended USABILICS for identifying these three types of wrong actions. Additional functionalities have been added to the system in order to keep detailed information concerning aborted interactions and to register the time taken to perform each subsequence of a given task. With these extensions, it is possible to identify the subsequences of a task in which end users have trouble. Moreover, it is possible to identify objects and containers associated to each problem and, therefore, provide detailed recommendations in order to suppress or, at least, minimize the problem.

Towards identifying relations among wrong actions and concepts of the COP model, we performed experiments with two Web applications: (I) an e-learning system and (II) a Web site for publishing information technology articles. During 52 days we monitored four tasks, two from each application. As a result, we gathered from application I a total of 1,019 attempts to perform tasks, which resulted on 258,802 log entries, and from application II, a total of 1,605 attempts and 474,437 log entries. Following, we detail the tasks defined in applications I and II.

Tasks defined for application I:

1. Correcting a student's assignment
   (a) Select the option *Correct* in main menu
   (b) Select a course for correcting assignments and click on the link *Correct*
   (c) Choose a student's assignment and click on the link *Correct*
   (d) Assign a grade and fill up a comment
   (e) Click on the button *Correct* to send the grade and the comment to the student
2. Sending a mail message for a group of students
   (a) Select the option *Students information* in main menu
   (b) Select a classroom
   (c) Check the target students (the ones that will receive the message)
   (d) Fill up the text fields *subject* and *message*
   (e) Click on the button *Send email*

Tasks defined for application II:

1. Sign up to receive articles
   (a) Click on the link *Sign up to receive articles*
   (b) Fill up the text fields *name* and *email address*
   (c) Click on the button *Sign up*
2. Sending a comment to the author(s) of an article
   (a) Select the option *Comment* in main menu
   (b) Fill up the text fields *name*, *email address*, *subject* and *message*
   (c) Click on the button *Send email*

Comparing the captured tasks to the corresponding defined tasks, we observed a correlation between low usability indexes and the presence of wrong actions. USABILICS identified patterns of wrong actions associated to hyperlink clicks, to the opening of new pages, to the scrolling in pages and to the interaction with forms. These patterns were classified in six categories:

1. Clicks in links that do not belong to the optimal path of the task;
2. Opening of pages that do not belong to the optimal path of the task, before accessing a link;
3. Opening of pages that do not belong to the optimal path of the task, before accessing a form;
4. Excessive scrolling before filling a form;
5. Excessive scrolling before accessing a link;
6. Large time interval between requesting a page and the page load event.

Whenever one of these patterns is found, we identify the interface element which is most likely involved with the corresponding problem. Following, we provide a recommendation that usually requires an update on the identified object. The proposed recommendations are enumerated from R1 to R7, in such a way that, R1 is the recommendation for pattern 1, R2 is for pattern 2, and so forth.

R1. Highlight the link X from other links on page P. Use icons.

R2. Change the location of the link X on page P. Put it on a page that is frequently accessed by users.

R3. Change the location of the form F on page P. Put it on a page that is frequently accessed by users.

R4. Bring the form F to the top of page P. Identify the form with an adequate title.

R5. Highlight the link X on page P. Try to put the link on the top of the page.

R6. Optimize the loading of page P.

3.1 Rationale for recommendations

In order to examine the rationale behind the recommendations produced by USABILICS, we refer again to the evaluation of tasks in our experiment. For task 1 in application I, it was found that only 16% of end users that started the task were able to accomplish it, resulting in a very low usability index (0.1033). USABILICS pointed out that 36% of end users have aborted the task in subsequence (b) and the same percentage have aborted in subsequence (c). In both cases, the system identified that the wrong actions performed by users were related to scrolling, what indicates that users might have been experiencing difficulties on finding an interaction element. Therefore, the system chose recommendation R5: *Highlight the link "Correct" on page "index.php". Try to put the link on the top of the page.* For task 2 in application I, the system identified that end users have taken on average 11 seconds to click on the link that starts the task, pointing out that the link *Students information* is not highlighted from other links, therefore recommendation R1 have been given. Moreover, in subsequence (b) of this task, it was noticed excessive scrolling associated to the filling of a form. As a result, recommendation R4 was also displayed for this task. The usability index for this task was 0.3757. Only 35% of end users were able to accomplish the task.

The computed usability index for task 1 in application II was 0.5359. For this task, the completion rate was 100%, however, it was found wrong actions related to clicking and scrolling in subsequence (a) and to scrolling in (b). Based on this findings, R5 – *Highlight the link "Sign up to receive articles" on page "article20.php". Try to put the link on the top of the page* – and R4 – *Bring the form "SignUp" to the top of page "article20.php". Identify the form with an adequate title* – were recommended. Only 50% of end users were able to accomplish task 2 in application II. As a result, this task presented an usability index of 0.3217. The low value of the index is also due to the large amount of wrong actions found on subsequence (b) of the task – mostly related to the filling of a form. About 47% of the wrong actions were related to scrolling and 31% were associated to clicking on non-target objects. Based on these findings, the systems recommendations were R1 – *Highlight the link "Comment" from other links on "index.php". Use icons* – and R4 – *Bring the form "CommentForm" to the top of page "index.php?area=comment". Identify the form with and adequate title.*

3.2 Validating the recommendations

In order to verify the quality of the provided recommendations, we asked developers to update the tested applications according to the suggestions provided by USABILICS. After the updates, we performed the same experiments. The obtained results in this second round of tests are as follows. For task 1 in application I, the usability index has risen to 0.4679 and the wrong actions previously found did not appear at this time. However, the system identified a new problem related to the loading of the page that shows the list of courses taught by a given teacher, and recommended R6. For task 2 in application I, the updates were also effective: the new usability index was 0.5759. The system pointed out that wrong actions related to scrolling reduced 99% as a result of the recommendation R4. Similarly, the implementation of recommendation R1 also solved the problem related to the wrong actions performed before the click on the link "Students information". Actually, the amount of wrong actions detected in this subsequence of the task has reduced 62%.

The usability index for task 1 in application II has increased to 0.6807. The performed update reduced 88% the number of wrong actions. Besides the good results in terms of both the usability index and the quantity of wrong actions, we observed that the number of registration of new users has increased after the update. In our opinion, this reflects the fact that the redesign has been an important factor in order to attract more users to the Web site. Task 2 in application II has also presented improvements after the update: the usability index has risen to 0.6180 and the wrong actions associated to clicking in non-target links were not detected anymore. Moreover, the average time for starting the task has reduced in 3 seconds, validating, therefore, the effectiveness of the recommendation R1. Updates related to the positioning of the commentary form, as suggested in recommendation R4, were also effective towards reducing the wrong actions associated to scrolling. With the implemented improvements, the mean time to accomplish the task has dropped 25%. In spite of these good results, about 20% of the users were not able to fully accomplish the task, what explains the usability index of 0.6180.

4. RELATED WORK

Remote and automatic or semi-automatic usability evaluation is an important tool in order to support the development of modern Web applications. According to Ivory and Hearst [6] the automation of usability evaluation increases the coverage of evaluated features and, at the same time, reduces the cost of the evaluation process, as (i) the time needed to perform usability evaluation decreases significantly, and (ii) the need for expert evaluators is reduced or even eliminated. Tullis et al. [13] emphasizes the advantages of remote usability testing, when compared to laboratory-based testing. According to them, both lab and remote tests capture very similar information about the usability of a site. However, remote evaluation is much more cost-effective than lab tests.

The most used approach for performing remote usability evaluation is based on capturing the end user interaction using logs. WELFIT [11], UsaProxy [2], WAUTER [3] and WebRemUSINE [7] are examples of tools that exploit this approach. WELFIT employs a graph-based algorithm for identifying patterns on the events performed by the end user in a single page. As a result, it presents a chart that summarizes the most important actions that have been performed. Comparing to our system, this tool do not consider paths connecting different pages and do not provide specific information concerning usability problems.

UsaProxy is a proxy-based system in which the capture of events depends on the configuration of the user's browser

and the usage of a proxy-server. As USABILICS, this system performs the analysis of tasks, however, it identifies elements during the definition of tasks solely based on the *id* attribute of the HTML language. Therefore, it is not capable of treating elements generically, as it is the case in USABILICS.

WAUTER also performs task analysis in its evaluation process. For defining tasks, WAUTER exploits a specif task modeling notation and uses pre-defined heuristics for comparing the actions performed by end users to these tasks. Differently from USABILICS, the approach for the definition of tasks in WAUTER is complex and time consuming. WAUTER also presents limitations for the definition of alternative paths for a given taks, specially when dynamic Web applications are considered.

WebRemUSINE, as USABILICS, relies on a task model to perform the analysis of tasks. However, for defining tasks in WebRemUSINE, evaluators need to use a special editor and a table for mapping entries in the log to defined tasks. Moreover, end users need to specify which task they are performing in order to associate the resulting logs to the correct task. Therefore, WebRemUSINE is not pervasive to end users and requires a lot of effort from evaluators. USABILICS, on the other hand, aims at providing usability evaluation without burdening evaluators and end users.

The literature also reports on systems that perform remote usability evaluation without gathering log data. UserZoom [15] and UserTesting.com [14] are examples of such systems. Towards evaluating an application, these tools provide a set of questions that end users are supposed to answer during their interaction with the application. This approach usually requires the support of expert evaluators in order to analyze the collected answers. Another drawback is that it requires an extra effort from the end users.

5. CONCLUSIONS AND FUTURE WORK

The literature reports on several tools for the remote and automatic or semi-automatic evaluation of Web applications. These tools, however, present limitations, specially for the evaluation of large and dynamic applications. In order to tackle the limitations of these tools, we developed USABILICS. The main goal of our system was performing usability evaluation without burdening both the application developer and the end user. Towards this goal, we proposed COP, an interface model that supports the generalization of similar tasks. Based on COP, we developed a tool that allows the definition of tasks in a simple and intuitive way, by simply interacting with the application's graphical interface. This same tool is used for gathering the end user interactions, generating logs that, finally, are compared to the previously defined tasks. This comparison provides a metric for the efficiency and for the effectiveness of each evaluated task. We called this metric the usability index.

In order to validate our system, we selected tasks from two different applications and evaluated them using USABILICS. Following, we applied laboratory-based tests to the same tasks. We noticed an agreement between the results of the lab tests and the values of the usability indexes [16]. Despite the good results, developers that participated in these experiments pointed out that USABILICS could be more effective if it provided recommendations in order to solve the detected usability problems. Towards answering this request, we extended USABILICS providing recommendations which detail actions to be performed to tackle specific usability problems. The experiments performed showed the effectiveness of our recommendations in order to raise the usability index of the evaluated applications.

We envision, however, some features that need to be improved in USABILICS. The approach used for defining tasks, for example, needs to be extended. As of today, our system supports the definition of tasks composed of linear paths, where the beginning and the end of a task is clearly defined. There are, however, tasks that do not present a linear sequence of events. Consider, as an example, the task *adding a product to a shopping cart*, in an e-commerce application. The end user may perform a linear sequence of steps in order to select a product and then add and remove the product to and from the cart several times. Just because the user is not sure about buying the product, it does not mean that she is performing a wrong action. USABILICS is currently unable to detect this kind of behavior. Therefore, as a future work, we plan to support the definition and analysis of complex and non-linear tasks. Besides supporting the repetition of sequences, we plan to support the definition of tasks that may be composed by subsequences of events which can be performed in any order. This is important, for instance, in order to consider the several correct possibilities of filling a form.

We also plan to exploit the interaction patterns and the usage context in order to identify user profiles. Based on these profiles, we plan to use a design patterns approach in order to apply successful solutions to applications that present similar user profiles.

6. ACKNOWLEDGMENTS

The authors would like to thank the Brazilian agencies FAPEMIG, CAPES and CNPq for the financial support.

7. BIOGRAPHIES

Leandro Guarino de Vasconcelos is a master student in Computer Science at Universidade Federal de Itajubá (UNIFEI). He is an assistant professor of information technology at Faculdade de Tecnologia de Guaratinguetá. He pursues a specialization in interactive media at Senac, SP. He is also a systems analyst at Faculdade de Engenharia de Guaratinguetá, UNESP. His research focuses on human-computer interaction and usability.

Laércio Augusto Baldochi Jr. is an assistant professor of Computer Science at Universidade Federal de Itajubá, where he leads the Usability and Interactive Media Lab. He is an advisor in the Computer Science Master's Program of UNIFEI. He received his PhD in Computer Science from Universidade de São Paulo. His research interests include ubiquitous computing, hypermedia systems, semantic Web and human-computer interaction.

8. REFERENCES

[1] M. S. Andreasen, H. V. Nielsen, S. O. Schrøder, and J. Stage. What happened to remote usability testing?: an empirical study of three methods. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1405–1414. ACM, 2007.

[2] R. Atterer. Logging usage of ajax applications with the usaproxy HTTP proxy. In *Proceedings of the*

WWW 2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection, 2006.

[3] S. Balbo, S. Goschnick, D. Tong, and C. Paris. Leading web usability evaluations to WAUTER. In *Proceedings of the Eleventh Australasian World Wide Web Conference*, 2005.

[4] Google. Google analytics. `http://www.google.com/analytics`, 2011.

[5] ISO/IEC. 9241-11 Ergonomic requirements for office work with visual display terminals (VDT)s - Part 11 Guidance on usability. Technical report, ISO, 1998.

[6] M. Y. Ivory and M. A. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.*, 33:470–516, 2001.

[7] L. Paganelli and F. Paternò. Intelligent analysis of user interactions with web applications. In *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, pages 111–118. ACM, 2002.

[8] F. Paternò and L. Paganelli. Remote automatic evaluation of web sites based on task models and browser monitoring. In *CHI '01 extended abstracts on Human factors in computing systems*, CHI EA '01, pages 283–284. ACM, 2001.

[9] D. Raggett, A. L. Hors, and I. Jacobs. The global structure of an HTML document. `http://www.w3.org/TR/1999/REC-html401-19991224/struct/global.html`, 1999.

[10] A. Rivolli, D. A. Marinho, and L. T. E. Pansanato. Wautt: a tool for tracking the user interaction in interactive web applications (in portuguese). In *Companion Proceedings of the XIV Brazilian Symposium on Multimedia and the Web*, WebMedia '08, pages 179–181. ACM, 2008.

[11] V. F. Santana and M. C. C. Baranauskas. Summarizing observational client-side data to reveal web usage patterns. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1219–1223. ACM, 2010.

[12] T. Tiedtke, C. Märtin, and N. Gerth. AWUSA: A tool for automated website usability analysis. In *PreProceedings of the 9 th International Workshop on the Design, Specification and Verification of Interactive Systems*, pages 251–266, 2002.

[13] T. Tullis, S. Fleischman, M. McNulty, C. Cianchette, and M. Bergel. An empirical comparison of lab and remote usability testing of web sites. In *Usability Professionals Association Conference*, 2002.

[14] UserTesting.com. Usertesting.com: Low cost usability testing. `http://www.usertesting.com/`, 2011.

[15] UserZoom. Userzoom: zooming in on the user experience. `http://www.userzoom.com/`, 2011.

[16] L. A. Vasconcelos and L. A. Baldochi. USABILICS: remote usability evaluation and metrics based on task analysis (in portuguese). In *Proceedings of the 10th Brazilian Symposyum on Human Factors in Computer Systems & 5th Latin American Conference on Human-Computer Interaction*, pages 303–312, 2011.

[17] A. P. Vermeeren, J. Attema, E. Akar, H. de Ridder, A. J. von Doorn, Ç. Erbug, A. E. Berkman, and M. C. Maguire. Usability problem reports for comparative studies: Consistency and inspectability. *Human Computer Interaction*, 23(4):329–380, 2008.