Customizable Automatic Detection of Bad Usability Smells in Mobile Accessed Web Applications

Fabio Paternò, Antonio Giovanni Schiavone, Antonio Conte
CNR-ISTI, HIIS Laboratory
Pisa, Italy
{fabio.paterno, antonio.giovanni.schiavone, antonio.conte}

ABSTRACT

Remote usability evaluation enables the possibility of analysing users' behaviour in their daily settings. We present a method and an associated tool able to identify potential usability issues through the analysis of client-side logs of mobile Web interactions. Such log analysis is based on the identification of specific usability smells. We describe an example set of bad usability smells, and how they are detected. The tool also allows evaluators to add new usability smells not included in the original set. We also report on the tool use in analysing the usability of a real, widely used application accessed by forty people through their smartphones whenever and wherever they wanted.

Author Keywords
Remote Usability Evaluation; Web Mobile application log analysis; Usability Bad Smells.

ACM Classification Keywords
H.5.2 User Interfaces: Evaluation/methodology

INTRODUCTION

The World Wide Web is an indispensable global means of communication for people, companies and public organizations. Building easy-to-use Web applications has become a crucial element for anyone who wants to promote services or convey information. This need is made even more acute by the widespread use of mobile devices, which although still supporting the Web have also changed the way people make use of applications. Currently, mobile devices are the most often used platforms to perform recreational activities (such as performing search queries [12] or enjoying multimedia content), and their usage in business activities is consistently increasing [8] as well.

For several years, researchers have conducted studies about the issue of analysis and improvement of Web application usability, proposing several tools, methodologies and techniques for this purpose. In particular, automatic usability evaluation tools [21] have been considered with the aim of reducing the time and costs involved in usability analysis, freeing evaluators from repetitive and tedious tasks, and allowing assessments to be scaled up without increasing the evaluation costs excessively. Automatic Web usability evaluation tools can be classified into two main groups: those that use Web pages' source code (i.e. their structure and/or content) as the data source for usability issue detection, and those that focus on actual user interaction data analysis. The first group includes some commercial tools such as Google's Mobile Friendly Test Tool [13] or Bing's Mobile Friendliness Test Tool [4]: starting with the structure of the Web page, these tools try to infer its usability in the specific context of navigation performed through mobile devices. For the latter group, usage data can be retrieved from server logs (mainly containing the chronological sequences of visited Web pages) or by client-side logging of users' activities while they are browsing (thus recording both the sequence of visited Web pages and the infra-page interactions, such as clicks, scrolling, etc.).

The rise of devices such as smartphones and tablets has led to the wide adoption of types of user interactions, which are significantly different from those on desktop devices. These differences arise from the many possible contexts of use, from technical limitations of mobile devices (e.g. connectivity, small screen size, different display resolutions, limited processing capability and power), and from the way in which users interact with them (for instance, some users prefer to interact with smartphones with both hands, others prefer to interact with a single hand [5]). Variations in each of these factors (e.g. the change of the screen size [28]) can therefore lead to different perceptions of usability.

In order to better understand and analyse these types of user interactions it is therefore necessary to define criteria and develop new evaluation tools to ensure proper usability evaluation even in such mobile contexts. In this paper, we present a method for automatic detection of usability issue indicators (also known as "Bad Usability Smells") in mobile access to Web applications. This method is supported by Mobile Usability Smell Evaluator (MUSE), a new proxy-based Web usability evaluation tool that is able to record user behaviour while interacting with any Web application through any type of browser-enabled device. The

MobileHCI '17, September 04-07, 2017, Vienna, Austria

ACM ISBN 978-1-4503-5075-4/17/09…$15.00
http://dx.doi.org/10.1145/3098279.3098558

identification of usability issues is carried out through an algorithm for identification of specific interaction patterns: recorded user interactions are compared with a repository of interaction patterns that indicate the potential presence of usability issues.

The paper is structured as follows. After discussion of related work, we introduce the concept of "bad smells", with special regard to those related to usability aspects. Afterwards, we define six different user behaviours that may indicate the presence of bad usability smells in mobile environments. Then, we describe our bad usability smell detection method and tool, illustrating the overall architecture, a language developed to represent the bad smells, and the detection algorithm that has been designed and implemented. Finally, we report on the results gathered through a user test carried out in order to evaluate the effectiveness of the usability evaluation tool, and then draw some conclusions with indications for future work.

RELATED WORK

The advent of mobile technology introduced new usability issues. Thus, usability evaluation methods need to be revised in order to address them. For example, one possible approach (Keystroke Level Model (KLM) [7]) is to define a model in order to predict user performance and usability issues, and it has been extended to estimate interactions on touch-screen interfaces as well [29].

Another approach is the analysis of the navigation paths followed by a user while performing some specific task in a Web site, and its comparison with the envisioned optimal navigation path for that task. The users' navigation paths can be obtained by server log analysis or logging client-side user interactions. One method using server logs for analysing user behaviour for evaluating Web site usability [11] compares users' navigational paths with optimal ones in order to detect possible usability issues. Unfortunately, server logs do not provide information related to users' behaviour within the Web pages, and hence information about which usability issues they may encounter while interacting with the Web page elements. One contribution on tracking user activity on Web Pages was UsaProxy [2]: in this tool, user activity recording was limited to the events generated via mouse or keyboard, and no processing or comparison between the recorded data was performed.

Navigation paths comparison is usually performed by applying some metrics. One of the metrics used for this purpose is a non-Euclidean distance measure called the Sequence Alignment Method (SAM) [18]. A solution that has considered SAM is WUP [6]. This tool allows the comparison between actual user behaviour and an optimal sequence of actions through a specific implementation of the SAM method. However, deriving usability issues from this SAM-based analysis has proven rather difficult. A similar tool is WELFIT [30], which uses a JavaScript that must be included manually, and registers user interactions as client-side event logs. By analysing the logs, the tool is able to identify recurring interaction usage patterns. The analysis is performed through a labelled digraph representing the user interactions. However, the detection method has shown to be able to detect only limited usage patterns. In addition, the tool was focused on monitoring usability in desktop environments, ignoring the mobile ones. Lettner et al. [22] have proposed another approach: they developed a solution for automatically extracting and grouping interaction sequences from users in mobile environments. This solution requires the development of mobile apps through the use of a framework able to annotate the source code and thus define some apps' states already during the development phase. Thus, the proposed framework only automates clustering and classification of interaction sequences, but it does not provide functionalities for automatic detection of suspected usability issues.

In recent years, a different approach has started to be considered in the field of usability evaluation: the main idea is to define and formalize structures, user behaviours and other types of anomalous data that serve as clues ("Bad Smells") for possible usability issues, and verify their potential presence. The concept of "Bad Smell" was proposed by Fowler and Beck [9] and it comes from the field of source code refactoring. The main idea is that an expert should be able to "to look for certain structures in the code that suggest the possibility of refactoring". In this regard, some metrics have been defined, for example in Java source code [23]. The usage of refactoring techniques has been extended not only in order to improve code quality, but also to improve other aspects such as usability. A proposal to use the concept of "Bad Smell" to enhance the usability of Web applications [10] suggested utilizing refactoring techniques not only to improve "internal quality attributes" (for instance, database performance) but also to improve external quality attributes, including usability. These authors proposed a first categorization of "Bad Smells" in two broad groups, "Navigation and Presentation". An attempt to automatically detect "Bad Usability Smells" (i.e. "Bad Smells" underlying the presence of usability issues) [15] consists in a detection tool composed of three different modules. The Threats Logger is able to register higher-level interaction events and to process them to generate a list of "usability threats". The Bad Smells Finder is a server-side application that receives usability threats and stores them for analysis, and the Bad Smells Reporter is a module able to retrieve the stored threats and displays the resulting bad smells. Unfortunately, this approach requires manual script installation, is based on fixed a priori assumptions in terms of possible threats, and is not particularly structured. Recently, the same authors proposed a similar tool [16], which also allows a "real time" reporting of detected usability smells. Unfortunately, in certain circumstances this features can lead to a "flickering" reporting (i.e. during the flow the user interaction, the tool can temporarily and mistakenly recognize parts of user behaviour as a usability

smell), and it does not allow to visually analyze the user interaction.

A similar tool for automatic detection of "Bad Usability Smells" is AutoQUEST [17] [19], which also records user actions through a JavaScript that must be included manually. Afterwards, the data are processed in order to generate Task Trees representing the recorded user interactions. The analysis of the differences between the expected occurrences and those generated in the Task Trees can lead to the detection of four distinct Bad Usability Smells: “Missing feedback”, “Important task”, “Required inefficient actions” and “High website element distance”. However, in this case a manual script installation is required, and the definition of Bad Usability Smells is fixed. Moreover, the generation of Task Trees can be a time-consuming process, and does not allow usability experts to perform real-time analysis. Another example of using task models to support analysis of logs interactions was presented in [26], in which the task model represented how the tasks were expected to be accomplished and the logs the actual performance, so that deviations from the task model pointed to potential usability issues. Task models can provide compact abstract representations of multiple optimal logs but require some experience in formal modelling that many usability evaluators may not have.

W3Touch [24] is a tool that aims to support adaptation by dynamically modifying the Web page. Firstly, user interaction data are collected through a JavaScript injected in the Web page, then visualization techniques are applied to the recorded data in order to segment the interface and identify critical components, and finally designers can test different adaptations and indicate the adaptations to be applied in specific contexts. One limitation is that the tool metrics are limited to detecting only: mis-clicks (i.e touches that miss an intended target) and elements that need to be zoomed (because they are too small or too close to other elements). The focus of W3Touch is different from ours: it aims to define adaptation rules for the dynamic modification of the structure of the Web page according to some predefined metrics instead of carrying out an analysis of the Web application usability. Furthermore, extensions of the set of adaptation rules in W3Touch requires implementing new callback handlers (i.e. adding new code), while we propose a more flexible and less code-dependent approach for extending our tool: adding the analysis of a new bad smell just requires its definition in the XML-based specification. Lastly, another possible approach in this area is performing long-term Web application monitoring in order to extract micro behaviours [1]. However, this approach requires prolonged effort over time and has problems identifying different accesses by the same users over time, thus its application is still in its early stages.

To summarise, the application of the concept of bad smell in the field of usability evaluation is an emerging and promising methodology, which could simplify and improve the automatic detection of usability issues in Web applications. Nevertheless, nowadays a solution that comprehensively exploits the bad smell-based approach for usability evaluation of mobile applications with the possibility of adding new smells without changing the tool implementation is still lacking. Our proposal aims to fill this gap.

BAD USABILITY SMELLS

Introduction

As previously mentioned, “bad usability smells” can be viewed as clues about the presence of some usability problems within an interactive application. In this approach, the users themselves are the first actors involved in the process of usability problem detection, since their interactions are the source generating the bad usability smells. Detection of user interactions should be carried out in an unobtrusive manner in order not to affect users’ behaviour while performing their tasks.

The strategies adopted to accomplish tasks can vary from user to user according to various aspects such as different contexts of use, different devices, different personality etc. Even faced with the same usability problem, users may adopt various strategies (and consequently, user behaviours) to overcome it. Nevertheless, often there is a small subset of behaviours that is most frequently adopted by users to cope with that specific usability issue: this subset is thus the basis for the definition of bad usability smells. Indeed, various studies have been carried out aiming to capture such smells through different methods: for example, [16] and [17] have focused on how to use them with desktop applications, while others have considered the identification of such behaviours for a different purpose, such as supporting adaptation [24].

Defining Bad Usability Smells

In mobile devices, there are some issues caused by the limited screen size and other touch input issues because of pointing accuracy [20], thus users can encounter various problems if the user interface design does not consider these aspects. A first phase of our work was to establish a set of usability issues that can be revealed by user behaviour. To identify this set we have relied on data presented in related work (e.g. [15], [24] and [26]), information provided by commercial software such as [14], significant studies regarding mobile usability [25], and our analyses of various Web sites and how people interact with them through mobile devices. At the end of this phase, six different usability issues have been identified, and are described in the following.

Too Small or Close Elements: this smell is characterized by the presence of Web page elements that are excessively close to each other. Too closely positioned form fields, or too small and / or insufficiently spaced selection buttons are some examples. To view the content effectively, the user is forced to perform a series of complex actions in order to resize the content.

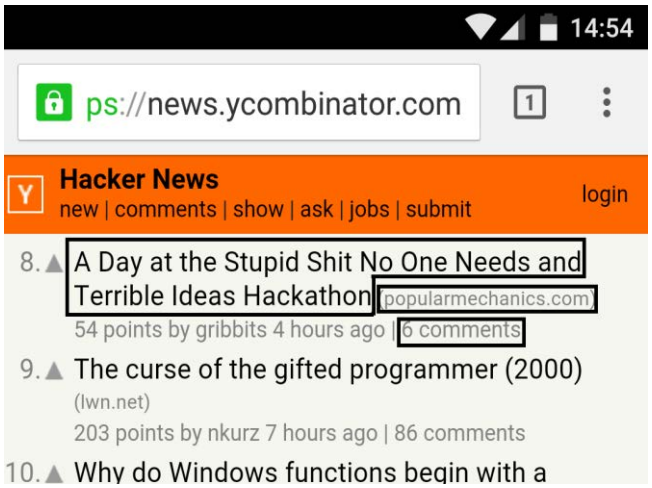

Fig. 1. Example of Too Close Links Bad Usability Smell

Too Close Links: This problem represents a variant of the previous case, but it is identified by different user performance: in fact, it involves the presence of very close elements whose interaction activates the loading of a new Web page. An example is a series of links with reduced line spacing or the submit form button insufficiently distanced from the other elements. In this case, the users may mistakenly load another Web page, and are forced to retrace their steps. Figure 1 shows an example in which the three links highlighted by black rectangles are so close that wrong touch selections can easily result.

Distant Content: this problem regards the presence of related Web page contents arranged too far from each other,

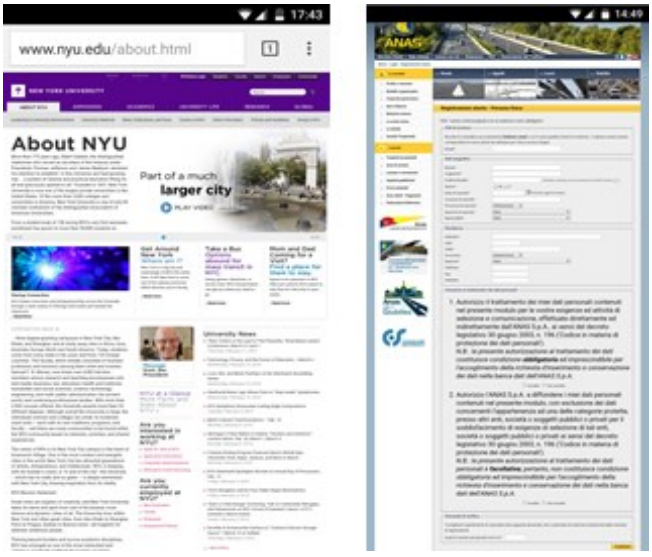

Fig. 2. Example of Bad Readability (left) and Long Form (Right) Bad Usability Smells

and whose display or interaction is crucial for the proper execution of some tasks. The user is forced to perform a high number of upward and downward scrolls.

Too Small Section: a section whose size appears too small requires specific magnification actions. The corresponding expected behaviour is to enlarge the section through a double tap (a gesture that in many mobile devices works as a shortcut for zooming), and then continue the activity. These actions allow users to facilitate task performance.

Bad Readability: This smell regards the difficulties encountered by the user during reading text content. This case is detected when interacting with blocks of text whose font size is too small or the spacing is too low. Figure 2 left shows an example. The user takes a series of actions aimed at optimizing the font size and / or location of the text block for ease of reading, such as sequences of pinch and pan actions but not followed by interactions such as taps because the task is reading.

Long Forms: This smell is characterized by the presence of a high number of interactions with input fields, considered excessive for the purposes of good usability on mobile devices. An example is in Figure 2 right.

MOBILE USABILITY SMELL EVALUATOR

Architecture Overview

Mobile Usability Smell Evaluator (MUSE) is a new proxy-based Web usability evaluation tool that is able to record the behaviour of a user while interacting with any Web application through either desktop or mobile devices.

The data on user behaviour are collected through a JavaScript logger injected into the Web page through a proxy server: thus, the tool is able to record user interactions on any Web site, and therefore without the need for the owner of the Web site to manually install the data logging scripts (see Figure 3).

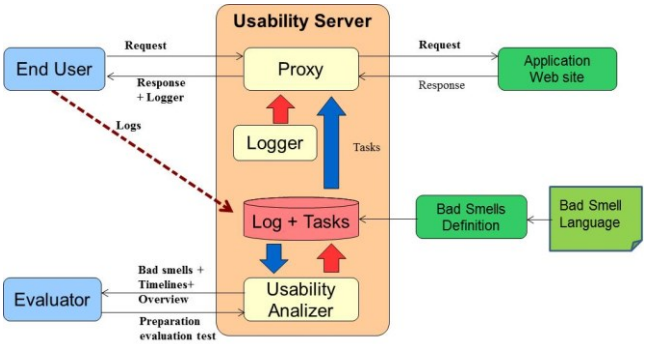

Fig. 3. The MUSE Architecture

The proxy also includes a panel used to indicate the tasks to perform in that application by the users at the beginning of the test session. Each user interaction is recorded as a sequence of events that are generated directly by the user (e.g. tap, pinch, mousemove, click, etc.) or the browser in response to user actions (e.g. page resize, mobile device orientation change). Indeed, our solution is able to analyse all typical events of touch-based mobile devices (tap and

double tap, pinch, pan, swipe, press, rotate, orientation changes) and, for those events for which such information is relevant, also their direction (e.g. pinch out, pan down, swipe left). In order to detect typical mobile device events, our tool's logger exploits the functionalities offered by the Hammer.js library, which is able to detect such. The tool is also able to gather information from the smartphone sensors. For instance, if the user is performing a task through a device equipped with GPS, the logger will ask to share its localization: if the user agrees, geo-localization changes will also be recorded. Moreover, if GPS is not available or its access is not granted, our tool can access the mobile device accelerometer in order to infer whether the user is standing or walking.

The preparation of usability tests is carried out through the tool backend: usability experts can create and delete usability evaluation sessions and indicate the tasks that compose them. The tool includes a usability analyser module, which takes the data collected during user tests, and provides overview information on the collected logs (e.g. statistics on browsers, operating systems, and devices accessed by the users involved in the test), associated interactive timeline visualizations, and indications where the bad smells defined have occurred in such logs. Timelines deriving from different user behaviour logs for the same task can be overlapped in order to graphically compare the two distinct behaviours. It is also possible eventually to share access with other experts using the tool.

The data are collected anonymously and stored in a database: subsequently, they can be processed to highlight meaningful information on all the recorded interactions and possible user behaviours that seem to indicate usability issues (i.e. bad usability smells). For each event detected by the logger, the following information is recorded:

- Event type and, if meaningful, event direction (e.g. Pan left).
- Event Timestamp.
- If meaningful, the HTML tag on which the event was triggered and its identifier or, if missing, its XPath.
- If meaningful, the coordinates of the point on the screen where the event has been triggered.
- Other information depending on the event type (e.g. characters typed through the physical or virtual keyboard, GPS coordinates, path of the screenshot of the loaded page, etc.)

In addition, usability experts have the possibility to define some custom events (e.g. a click on a particular button or the transmission of data collected through a specified form), which have particular relevance for the usability analysis in a particular case, and which will subsequently be recorded by the tool in addition to the predefined events.

Bad smell representation

To define within MUSE a functionality for automatic detection of bad usability smell, a necessary step was to create a language to specify them. Since user sessions are recorded as sequences of events, even the bad usability smells had to be formalized as events patterns. Thus, the detection of bad usability smells is obtained by checking, within the recorded event sequences, the presence of event patterns that represent one or more bad usability smell.

During the development of the smell detection functionality, we noticed that it is very difficult to detect exactly identical subsequences of events. The user behaviours can vary in many, even minimal, ways, so it is impossible to associate a usability issue with an exact sequence. This aspect led to the introduction of a series of parameters, intended to facilitate the detection of similar sequences between them. In order to introduce some flexibility in the detection mechanism, we have introduced, in addition to the type of event, a set of parameters, to facilitate the detection of similar sequences:

- Number of repetitions: the number of consecutive occurrences of a specified event. When specified, this parameter can represent a minimal number of occurrences of the considered event (e.g. at least 5 times), or a quantity not defined. In the latter case, an indefinite number of repetitions is indicated by the symbol "*". If it is not indicated, then it means that the event should occur once.
- Direction: if relevant, it indicates the direction of the event in question. (e.g. It can be “up” for pan and scroll events, “out” for pinch events). Also in this case the value of the direction can be optional, indicating this situation with the symbol "$”
- Interval: define a time threshold that must be respected. The threshold represents the maximum time that elapses between the previous and the current event.

To clarify the concept, let us consider the previously defined “Too Small or Close Elements” bad usability smell.

Bad Usability Smell	Behavioral Pattern
Too Small or Close Elements	[*] Pinch(out) + [*]Pan($) + Tap + Focus(in)
Too Close Links	[*]Tap + Beforeunload + Pageview + Beforeunload + Pageview
Distant Content	[5]Pan(down)
Too Small Section	[*]Doubletap + [*]Resize + [*]Pan($) + [*]Tap
Bad Readability	[*]Pinch($) + [*]Pan($) + [3]Pan(down)
Long Forms	Tap + Focus(in) + [5]Focus($)

Table 1. Behavioral Pattern for each defined Bad Usability Smell.

In the case of this bad usability smell, we expect that the user will act in the following manner:

- perform an undefined number of pinch out (to zoom in the Web page).
- perform a series of pan events (for placing the interaction object at the center of the screen).
- finally perform a tap and consequently trigger a focusin event (to select / interact with the interaction object).

This behaviour can be translated in the following events subsequence:

[*] Pinch(out) + [*]Pan($) + Tap + Focus(in)

Likewise, the "Too Close Links" bad smell, which is associated with the presence of very close elements whose selection activates the loading of a new Web page, may cause that users mistakenly load another Web page, and are forced to go back to look for the right link. In terms of events sequence, this implies that the user mistakenly taps on a wrong link (Tap event), consequently the browser unloads the current page (Beforeunload event) and loads the new one (Pageview event). Then, the user will recognize that the wrong page has been loaded and return to the previous one by using the "back" button of the browser. Thus, the browser will unload the mistakenly loaded page (Beforeunload event) and reload the original one (Pageview event). Table 1 shows the event patterns associated with the defined Bad Usability Smells. In order to formalize and store the representations of the other defined Bad Usability Smells, we have chosen to utilize XML. An example of pattern formalization is shown in Table 2.

```
<?xml version="1.0" encoding="UTF-8" ?>
<patternsContainer>
  <pattern>
    <patternName>TooCloseElements</patternName>
    <event>
      <eventTitle>pinch</eventTitle>
      <direction>out</direction>
      <repnumber>*</repnumber>
      <interval>PT1S</interval>
      <targetElement></targetElement> </event>
    <event>
      <eventTitle>pan</eventTitle>
      <direction>left</direction>
      <repnumber>*</repnumber>
      <interval>PT1S</interval>
      <targetElement></targetElement> </event>
    <event>
      <eventTitle>tap</eventTitle>
      <direction>$</direction>
      <repnumber>1</repnumber>
      <interval>PT2S</interval>
      <targetElement></targetElement> </event>
    <event>
      <eventTitle>focus</eventTitle>
      <direction>in</direction>
      <repnumber>1</repnumber>
      <interval>PT1S</interval>
      <targetElement></targetElement>
    </event> </pattern> <pattern>
    ....
  </pattern>
</patternsContainer>
```

Table 2 Example of pattern definition

The advantages in using XML to formalize such patterns are that their description is easily understandable by both humans and computer systems, it can be easily extended and / or modified and can be validated by defining an appropriate XSD schema.

Smell detection algorithm

The problem of detecting the presence of bad usability smells can be reformulated as follows: given a sequence of elements (in our case event logs), verify if any of their subsequences correspond to a description of one of the bad smells defined in XML.

The problem is conceptually similar to the pattern matching performed using regular expressions (regex), with the following differences: regular expressions operate on a finite set of elements, while in our case the possible values in the logs can be infinite because the time component (timestamps) is defined by real values. Moreover, regular expressions only operate on the sequential position of the individual elements (characters / letters), while in our case our solution also evaluates the relationships between the temporal dimension of two adjacent sequences (that is, the interval between two events must not exceed a certain value).

Due to these differences, we have defined an algorithm that is based on the following steps: select all events in the logs that can be the beginning of one of the subsequences of interest (seeds), and from each of them start to build a "candidate" subsequence, verifying step-by-step the consistency with the pattern description (germination). If a "candidate" subsequence is incompatible with the pattern description, then it is eliminated (eradication), thereby reducing the set of "candidate" subsequences.

The algorithm depends on two preconditions: the pattern description of the subsequences always starts with an event of a specific type; each log is associated with a chronologically progressive numeric index.

The algorithm works in the following manner:

1. For each recorded interaction, it partitions the events that compose the sequence based on their type.
2. It selects the type of event indicated by the first element of a pattern description, and verifies whether the next events are compatible with the number of repetitions for the first event specified in the pattern description. The resulting elements are the "seeds" from which to build their candidate subsequences.
3. For all other elements of the pattern description:
   - 3.1. If it relates to an undefined event type, then there is no particular constraint, and the procedure moves to the next element.
   - 3.2. If it relates to a defined event type, it filters the partition by such type. Then, for each event in the logs considered:
     - 3.2.1. It verifies whether it follows immediately after the last element of a candidate subsequence.

3.2.2. It verifies if the event meets the criteria expressed by the pattern element in question.

3.3. In case there is one (or more) events satisfying the tests in point 3.2, they are added in the queue to a suitable subsequence, which is thus confirmed as a potential candidate.

3.4. The candidate subsequences referred to in the current iteration for which compatible patterns have not been found are eliminated, thus reducing the number of candidate subsequences

4. After all iterations, the candidate subsequences remaining are those that indicate the presence of a bad usability smell.

The bad smell detection algorithm can be applied to any log immediately after the end of the recording session.

Bad smell visualization

MUSE provides usability experts with a backend where they can graphically navigate the stored data in order to reconstruct the recorded user interactions. Each user interaction is represented as a sequence of events that have been generated directly by the user (e.g. tap, pinch, mousemove, click, etc.) or the browser in response to user actions (e.g. page resize, mobile device orientation change).

Each sequence is graphically represented as a timeline [27]: every event constituting the sequence is graphically represented by its own box containing the event type and an icon that illustrates in a simple way both the event type and, when meaningful, the event direction. Figure 4 shows an example of how a log is visualized as a timeline, and how a bad smell is highlighted in it.

Furthermore, the box for each event of type "Pageview" (i.e. an event triggered after loading a new Web page) includes a miniature of the loaded page: clicking on it brings up a screenshot of the page.

This is an important feature in the design of our timelines since it enables the usability expert to have a better understanding of the actual user interface accessed at that specific time, and thereby better analyse the corresponding sequence of events performed.

During the analysis, usability experts may have the backend highlight the bad smell detected, selecting both the bad smell type and the set of recorded user interactions to consider. The events which led to the recognition of one or more bad smell are highlighted in red (as shown in Figure 4): hovering over the highlighted part of the time axis brings up a small tooltip indicating which potential usability issue has been detected.

In our example the highlighted sequence suggests that the user had problems in the interaction with the Web page due to elements being too small or too close to each other, thus finding the need to zoom the page. Then, by clicking on the bar underlying the highlighted events, it is possible to call a popup that shows these events positioned directly on the screenshot of the page, visually reproducing the user interface design responsible for the usability issue. This functionality allows the tool to make even clearer the part of the user interface that led to the bad usability smell (as shown in Figure 5).

EMPIRICAL FEEDBACK

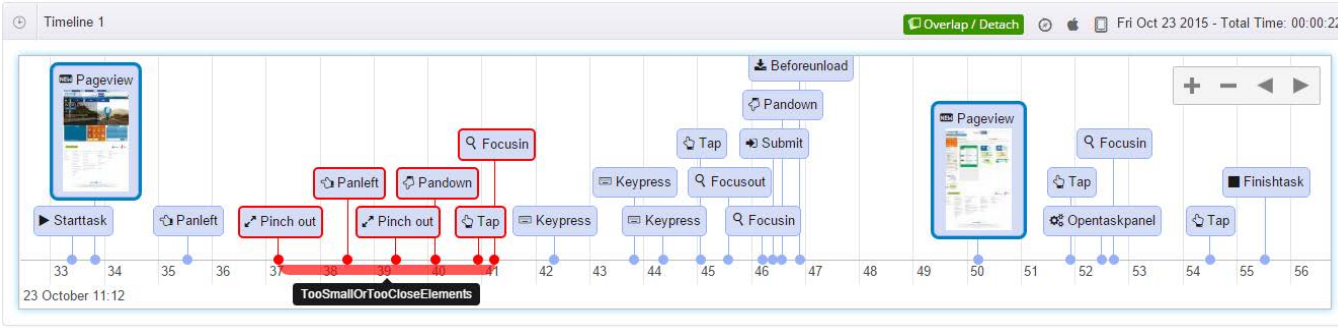

Fig. 4. Bad Usability Smells in MUSE's backend

In order to provide an initial validation of our method we have considered a case study involving a widely used Web site not developed by us. We first asked forty people to access it to perform some indicated tasks through their smartphones whenever and wherever they wanted. Then, we showed the results provided by our tool to eight usability experts to assess whether they were useful in identifying potential usability issues.

User Test Participants and Methodology

In order to verify the effectiveness of our approach, we firstly created a usability evaluation session, composed of four different tasks, regarding the English version of the Web site of "Autostrade per l'Italia" [3], the company responsible for construction and maintenance of Italian motorways and state highways.

We chose this Web site for the user test because our goal was to focus on a real Web site supporting many users with some services of public utility and with various usability problems, as often happens. These are the cases in which our tool provides most useful support. Another requirement was that it provide not only static information but also some support for interactive tasks.

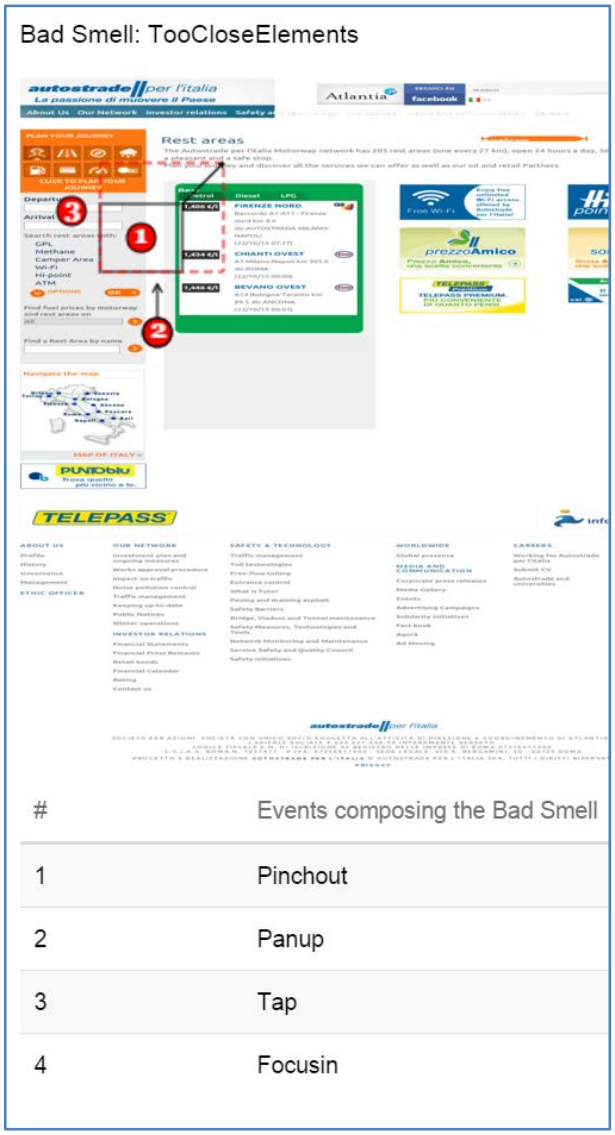

Fig. 5. Interaction that led to a Bad Usability Smell shown on the user interface

We asked users to accomplish four tasks:
1) Plan a journey from a particular city to another retrieving the best route,
2) Locate service areas along a specific route,
3) Retrieve today's weather information for another route,
4) Retrieve a gas station location along another route and check fuel prices.

Forty users performed the tasks by using their personal smartphones, thus generating a significant database of more than 14000 events. Each test was run remotely and each user was free to choose where, how and when to carry out the test. In addition, to ensure the privacy of each user, we deliberately did not record any personal information: the only information recorded were: Test Date; Test completion time; The type and model of the device used; The type and version of browser used. Since the test participation was promoted among some Bachelor students and their friends, we can estimate that the bulk of participants were aged between 25 and 35 years old. The devices used in the test were: 1 Windows Phone 8.1 device (Nokia Lumia 625), 1 BlackBerry Os device (Z10), 9 iOS devices (2 iPad and 7 iPhone), 29 Android devices (4 LG Optimus L5, 4 Samsung S4, 4 Samsung S3, 3 Samsung S3 neo, 2 Samsung S5, 2 Samsung S4 Mini, 2 Galaxy Note, 1 LG Optimus L70, 1 Galaxy Ace 2, 1 Galaxy Tab 3 8.0, 1 LG G3, 1 Sony Xperia Z3, 1 Motorola Fire XT, 1 Galaxy S Advance, 1 Google Nexus 6).

User Test Results

The tool was able to detect 51 instances of bad usability smells, distributed across the four proposed tasks, precisely 9 for task 1, 13 for task 3, 14 for task 3, and 15 for task 4.

In the first task, concerning the planning of an itinerary, the interaction was mainly focused on a search form in the home page. Seven users' logs contained usability issues, mainly related to the bad usability smell "Distant content", due to the difficulty users encountered in finding the exact location of items needed for the search. Moreover, one occurrence of "Too Close Links" and "Bad Readability" were also detected. The second task required users to search for specific service areas on the route: to complete the activity, users had first to access the specific section and then use a variety of search tools placed on the left side of the user interface. Due to the excessively small size of the search panel, a number of zoom actions in the left area were detected. The analysis detected seven occurrences of the "Too Small or Close Elements" bad usability smell. Moreover, there were two occurrences of "Too Small Section". Four instances of the bad Smell "Too Close Links" were detected as well, due to the extreme proximity of a set of icons on the home page, which are associated with various links to internal pages of the site, including the one for the search for service areas.

Figures 4 and 5 respectively show the "Too Small or Close Elements" bad usability smell for Task 2 on the timeline and highlight where the actions occurred in the user interface.

The third task required a search for information related to the weather in a specific motorway section: the structure of the search page was quite similar to that for the second task, and the results were quite similar, with a higher incidence of bad smells because of the presence of a drop-down list. In fact, for this task ten occurrences of "Too Small or Close Elements", three occurrences of "Too Small Section" and one of the "Too close links" were detected.

Finally, the last task proposed required a search for information related to a specific type of fuel: this was the most complex task and required the user to perform different actions (e.g., select the areas section service, enter data, tick the GPL checkbox). The results of the bad smell analysis pointed out 11 occurrences of "Distant Content". This issue stems from the presence of too many search results, which forced users to move around in the interface in order to

visualize all the desired information, thus causing excessive scrolling activity. In addition, two occurrences were detected for both the “Too Small Section” and “Too Small or Close Elements” bad usability smells.

The results of our tests showed that the most common bad usability smells in the considered application were “Too Small or Close Elements” and "Distant Content", identified with about the same frequency, and which together make up about 72% of the identified bad usability smells. In addition, the results showed a low presence or absence of "Bad Readability" and "Long Form" bad usability smells: this can be explained considering the Web application type and the tasks performed, which mostly focused on searching for short textual information obtained through the interaction with the forms.

Finally, as reasonable to expect, we found an increase in the number of the identified bad usability smells with increasing complexity of the tasks, from the nine identified issues for the first and simplest task, to the fifteen for the last and most complex task.

Usability Evaluators’ Feedback

Subsequently, eight usability experts were invited to use our usability evaluation tool in order to provide feedback on its functionalities and how it reports usability data. This expert group consisted of six males and two females, aged between 27 and 47 years (avg. 36.5 years, SD = 7.43). Furthermore, six of them had already had experience in the use of automated tools for usability evaluation. Each of them received a brief document describing the features offered by the tool and the credentials to access it.

The usability experts did not receive any particular task, but had the opportunity to freely use the tool, in complete autonomy and without any time limit, and to explore the logs database using the provided functionalities. They had the possibility to access all the data and analyses related to the user test reported in the previous sections.

From the tool-supported analysis it was possible to detect a number of usability issues in the various tasks. In particular, they referred to the presence of too close interaction elements, which forced users to zoom in and out, and the imperfect arrangement of content along the page, which forced users to repeatedly scroll upwards and downwards.

For example, Figure 6 shows the homepage of the Web site used for our test: the bottom of the central orange box contains a form with which users had to interact to complete the first proposed task. Due to the limited screen size, the two form fields turn out to be too close to each other, to the submit button and to the overlying link images (i.e. the credit card, gas station and speedometer icons). This poor design of the Web site generated a usability issue: this problem was revealed in the timelines by frequent sequences of pinch and pan events (to zoom in the page), or mistaken taps on one of the overlying link images and consequent loading of another Web page and quick return to the form page.

In the end, each participant was asked to complete a survey composed of 25 questions, relating both to a general assessment of the tool and to the provided functionalities: in 15 questions the experts could answer with a 7-point Likert scale rating (i.e. the higher score, the better the evaluation, as shown in Figure 7) and optionally provide suggestions, while

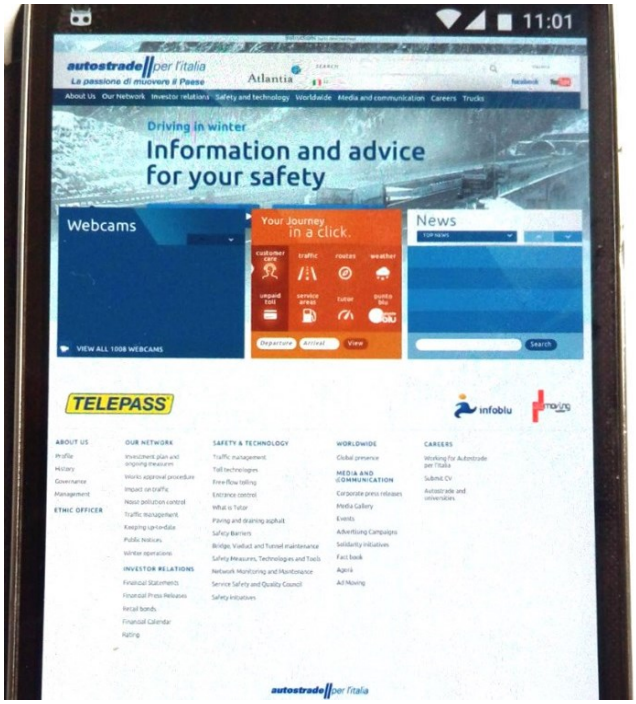

Fig. 6. Web site of “Autostrade per l’Italia”

for the remaining questions they simply agreed or disagreed with some statements regarding the tool.

The usability experts expressed an overall positive judgment about the tool’s graphical appearance and its clarity. Regarding the Overview view, the participants considered it extremely useful to have a general overview of user behaviour while performing a task and judged positively the choice of the data displayed in this view. Moreover, almost

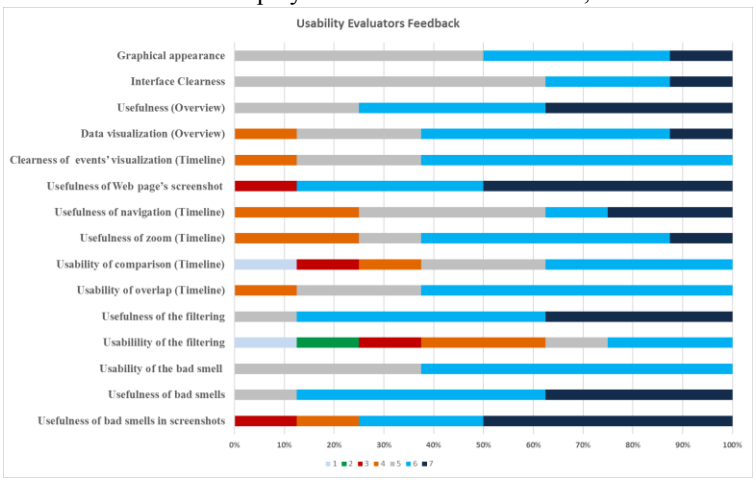

Figure 7 Usability evaluators feedback

all of them considered the data presentation to be sufficiently clear (87% of the group). Nevertheless, more than half of the participants (62.5% of the group) provided us with suggestions to improve the overview. They asked for more information in the task summary (the starting Web page's title, information about the total number of users who performed that task, the instructions provided to the user for the task execution) and several other ways to aggregate information about the sessions' duration.

The timeline view required a longer and more detailed investigation: experts considered clear and useful the events' visualization through timelines, found inclusion of the Web page's screenshot extremely useful, and perceived as usable both the navigation functionality and the zoom of the timelines. The majority of participants (87%) deemed the level of interactivity provided by timelines as sufficient, and suggestions were mostly limited to proposals to change the zoom icons (from +/- signs to magnifying glass-based icons). The topic that received the most discordant opinions was the timeline overlapping feature. In fact, all the experts considered visual comparison of timelines more effective than comparison based on quantitative data, and they also liked the opportunity to first overlap two timelines and then lock them so that they can move through them together along the time axis to inspect different periods of time. At the same time a minority of them criticized the usability of the timeline comparison method. These criticisms mainly regarded the possibility to place two timelines side by side instead of overlapping them: this functionality was considered more usable for comparing overly crowded timelines.

The experts also found the functionality of data filtering useful, although many requested the ability to choose the filtering logic, i.e. whether to remove the selected type of events or to leave them and remove the other types (87%). Moreover, regarding the bad smell detection functionality, the usability experts considered it both usable and useful.

All the experts also agreed that the representation of bad smells is clearly distinct from the events, and found the representation of the bad smells in the page screenshots useful. Thus, the feedback was positive regarding bad smell identification. We also collected some comments and suggestions, of which the most interesting were: the possibility of using different sizes and colours for representing different bad smells in the user interface screenshots; making the representation of the bad smells on the screen shot more interactive, in order to limit overlapping of numbers and arrows; introducing some documentation to interactively explain the bad smells, possible causes and remedies. We plan to address these suggestions in future versions of the tool.

CONCLUSIONS AND FUTURE WORK

We have presented a proposal for automatic detection of bad usability smells based on the analysis of data collected through a remote usability evaluation tool for mobile Web interaction. For this purpose, we have identified a first set of six common interaction patterns in mobile Web interaction that often correspond to usability issues of the considered application.

We have also designed and implemented an algorithm able to detect their presence in client interaction logs. Since the event patterns characterising the bad smells have been formalized through an XML-based language, our solution can be easily extended to consider further potential interesting bad smells. For this purpose it is sufficient to add in the dedicated XML file the definition of additional patterns through the language we have developed for this purpose and the tool will be able to detect them without the need of changing its implementation. We have also reported on a test of a real application by collecting and analysing data from logs obtained from the task performance of forty mobile users who accessed the application, not in a laboratory, but freely in the wild, wherever they pleased. The results have been useful to detect various usability problems in the considered application, which experts also found useful to support their analysis.

In order to optimize the performance, the underlying algorithm has been designed with the idea to progressively reduce the set of elements to process. Currently, the bad smell analysis is carried out server side as is the log data collection and processing as well. The entire process applied to the data set and bad smells reported in the paper takes about 40 seconds on a standard PC (Intel i3, 4gb ram). In such period of time about 25% is dedicated to the bad smell analysis. We are investigating further optimizations and performance improvements. In future work, we will also continue to improve the automatic bad usability smell detection by investigating whether there are other behavioural patterns to consider and further applying the proposed method.

REFERENCES

1. A. Apaolaza, S. Harper and C. Jay, "Longitudinal analysis of low-level Web interaction through micro behaviours" in Proc. the 26th ACM Conference on Hypertext & Social Media (Hypertext2015), 2015, pp. 337-340.
2. R.Atterer, M. Wnuk and A. Schmidt, "Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction" in Proc. of the 15th international conference on World Wide Web, 2006, pp. 203-212
3. "Autostrade per l'Italia" Web site. Retrieved February 6, 2017 from https://www.autostrade.it/en/home
4. Bing's Mobile Friendliness Test Tool. Retrieved February 6, 2017 from https://www.bing.com/webmaster/tools/mobile-friendliness
5. S. Boring, D. Ledo, X. Chen, N. Marquardt, A. Tang and S. Greenberg, "The fat thumb: using the thumb's contact size for single-handed mobile interaction" in

Proc. 14th international conference on Human-computer interaction with mobile devices and services (MobileHCI'12), 2012, pp. 39-48.

6. P. Burzacca and F. Paternò, "Remote usability evaluation of mobile web applications" in Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments, 2013. pp. 241-248.
7. S. K. Card, T. P. Moran, and A. Newell. "The keystroke-level model for user performance time with interactive systems." Communications of the ACM, vol. 23, no. 7, pp. 396-410, 1980.
8. ComScore's Global Mobile Report (July 14, 2015). Retrieved February 6, 2017 from http://www.comscore.com/Insights/Presentations-and-Whitepapers/2015/The-Global-Mobile-Report
9. M. Fowler, K. Beck, J. Brant, W. Opdyke and D. Roberts, "Refactoring: Improving the Design of Existing Code", 1999, Addison Wesley
10. A. Garrido, G. Rossi and D. Distante, "Refactoring for usability in web applications" in IEEE Software, vol. 3, no. 28, pp. 60–67, 2011.
11. R. Geng and J. Tian, "Improving Web navigation usability by comparing actual and anticipated usage," IEEE Trans. Human-Mach. Syst., vol. 45, no. 1, pp. 84–94, Feb. 2015.
12. Google's AdWords official blog (May 2015). Retrieved February 6, 2017 from http://adwords.blogspot.com/2015/05/building-for-next-moment.html.
13. Google Mobile Friendly Test Tool. Retrieved February 6, 2017 from https://www.google.com/webmasters/tools/mobile-friendly/
14. Google Search Console. Retrieved February 6, 2017 from https://www.google.com/webmasters/tools/
15. J. Grigera, A. Garrido, and J. M. Rivero "A tool for detecting bad usability smells in an automatic way" in Web Engineering, ser. Lecture Notes in Computer Science. vol. 8541 pp. 490–493, 2014.
16. J Grigera, A Garrido, JM Rivero, G Rossi, Automatic detection of usability smells in web applications, International Journal of Human-Computer Studies 97, 129-148, 2017.
17. P. Harms, J. Grabowski. (2014). Usage-Based Automatic Detection of Usability Smells. In Human-Centered Software Engineering, ser. Lecture Notes in Computer Science, 8742: 217-234.
18. B. Hay, G. Wets and K. Vanhoof "Mining navigation patterns using a sequence alignment method" in Knowledge and information systems vol. 6 no. 2, pp. 150-163, 2004.
19. S. Herbold and P. Harms, "AutoQUEST -Automated Quality Engineering of Event-Driven Software" in Proc. sixth IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2013), 2013, pp. 134-139
20. C Holz, P Baudisch, Understanding touch, Proceedings of ACM CHI 2011, 2501-2510, 2011.
21. M. Y. Ivory and M. A. Hearst "The state of the art in automating usability evaluation of user interfaces" in ACM Computing Surveys (CSUR) vol. 33 no. 4 pp. 470-516, 2001.
22. F. Lettner, C. Grossauer and C. Holzmann, "Mobile Interaction Analysis: Towards a Novel Concept for Interaction Sequence Mining", in Proc. 16th International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI 2014), 2014, pp. 359-368.
23. M. James Munro, "Product Criteria for Automatic Identification of "Bad Smell" Design Problems in Java Source-Code", in Proc. 11th IEEE International Software Criteria Symposium (CRITERIA 2005).
24. M. Nebeling, M Speicher and M. Norrie, "W3touch: metrics-based web page adaptation for touch" in Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI 2013), 2013, pp. 2311-2320.
25. J. Nielsen and R. Budiu. Mobile Usability. New Riders (2013)
26. L.Paganelli, F.Paternò, Tools for Remote Usability Evaluation of Web Applications through Browser Logs and Task Models, Behavior Research Methods, Instruments, and Computers, The Psychonomic Society Publications, 2003, 35 (3), pp.369-378, August 2003
27. F.Paternò, A. G. Schiavone, P.Pitardi. "Timelines for Mobile Web Usability Evaluation", in Proc. of the International Working Conference on Advanced Visual Interfaces (AVI 2016), 2016, pp. 88-91.
28. D. Raptis, N. Tselios, J. Kjeldskov and M. Skov "Does size matter? investigating the impact of mobile phone screen size on users' perceived usability, effectiveness and efficiency" in Proc. 15th international conference on Human-computer interaction with mobile devices and services (MobileHCI 2013), 2013, pp. 127-136).
29. A. D. Rice and J. W. Lartigue, "Touch-level model (TLM): evolving KLM-GOMS for touchscreen and mobile devices" in Proc. of the 2014 ACM Southeast Regional Conference, 2014.
30. V. F. de Santana and M. C. Calani Baranauskas "WELFIT: A Remote Evaluation Tool for Identifying Web Usage Patterns through Client-Side Logging" in International Journal of Human-Computer Studies, vol. 76 no. C pp 40-49, 2015.