10th IEEE International Conference on Software Testing, Verification and Validation Workshops

Pattern-Based Usability Testing

Fernando Dias
FEUP
fernandodias202@gmail.com

Ana C. R. Paiva*
FEUP / INESC TEC
apaiva@fe.up.pt

January 30, 2017

Abstract

Usability is a critical aspect of software systems because poor user experience can lead users to choose other software. One way to improve usability is through testing. But, usability testing is a challenge because, most of the times, it can not be accomplished without the presence of real users, which is complex and requires a lot of effort. However, there are some aspects and usability guidelines that can be tested automatically. This paper presents a test approach that defines generic test strategies (test patterns) to test usability guidelines (or best practices). It is an extension to previous work on testing functional aspects of web applications through the GUI (PBGT – Pattern Based GUI Testing). The main goal of this work is to be the first step in extending PBGT's PARADIGM language with usability testing patterns, so that it is possible to build test models from which usability tests can be generated and automatically executed over a website. This paper presents a new usability test pattern, called "Reachability Test Pattern", which is validated in a case study performed over an academic software system available on the web.

Keywords: Usability Testing, Pattern-Based Usability Testing, Test Patterns, GUI Testing, Software Testing.

I. Introduction

Graphical user interfaces (GUIs) have a great relevance in the interaction between the user and the platform used. There is a need to produce user-friendly and intuitive interfaces without the need to spend time learning how to use them.

According to the standard in [7], usability can be defined has 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". Thus, good usability should ensure that the system meets the goals for which it was built, making sure the user is satisfied with its use and preventing him from stop using it.

Testing systems' usability can take a lot of time and may need third party users, making it a complex and costly process. Although there are situations where the presence of real users is indispensable, there are also aspects of the usability tests that can be automated. The focuses of Pattern-Based Usability (PBUT) is on the usability tests that can be automated. PBUT extends previous work (PBGT - Pattern-Based GUI Testing [1][2][3][4][5]) in order to allow testing usability aspects on web applications. In order to do that, it defines a set of usability test patterns that can be applied to any website. So, PBUT has:

- An universal usability testing strategies that are able to test any website after a configuration step
- A configuration process to define input data to be used by the testing strategies to apply on any website
- An engine to execute the tests.

Besides the introduction this paper is composed of 4 more sections. Next section discusses automated usability testing approaches. The following section describes the algorithms and strategies used by the testing approach presented in this paper. Section IV presents

*Corresponding author

validation through a case study. Finally, some conclusions and future work.

II. Usability Testing

There are different platforms and tools to support usability testing. Some of them involve an analysis of user actions and browsing information (such as, mouse speed and mouse clicks). These platforms can be grouped into the ones based on user interaction analysis, A/B testing and based on metrics. Besides the platforms based on user interaction analysis there are:

- Platforms that perform automatic usability tests remotely, that is, register the activities of the user on a website in a non-intrusive way, with a subsequent analysis of those actions. Usually, they include a list of actions to be taken by the user and then analyze how the user executed them. As an example of this type of platform we have Usaproxy [8], m-pathy[1] and Web Usability Probe (WUP) [10];
- Platforms that apply Monte Carlo simulation and information retrieval methods to predict user behavior and navigation paths, aiming to accelerate and improve the design process of websites, such as WebTANGO[2];
- Platforms that record users mouse events or eyes movements and build maps of regions in which the users focuses their most popular click areas on a page such as Eye-Tracking[3] and CrazyEgg[4].

There are also automated platforms that perform A/B testing, which is a method that compares two versions of the same website and compares them with the purpose of determining which one is best in terms of usability. This type of test is done by providing questionnaires and usage scenarios, requiring the user groups to perform the tasks, and thereafter performing an evaluation of which of the available versions works best. Among these platforms are AttrakDiff[5], WaPPU and Optimizely[6].

There are still methods based on metrics that provide quantitative metrics for evaluating web pages. These are static metrics (based on HTML code) that do an analysis to detect some errors related to page design. Some examples of platforms that use these methods are: USherlock [17], which is a tool that is based on the structure of the interface, analyzing what users see on the screen, analyzing their static and dynamic aspects and W3TOUCH [16], which includes a set of specific metrics, related to the handling of the device that may impair the performance of the website in terms of usability.

The approach presented in this paper (PBUT – Pattern-Based Usability Testing) differs from all the approaches presented previously. PBUT does not require user interaction and neither performs static analysis for metrics collection. The goal of PBUT is to define generic test strategies that can automatically test the implementation of good usability practices in different web applications.

III. Approach

Pattern-Based Usability Testing is an extension to PBGT for usability tests. PGBT uses a Domain-Specific Language (DSL), PARADIGM [4], which allows to build models describing testing goals. The test cases generated from these models are then executed on a graphical user interface to find failures. PARADIGM is a graphical DSL with elements and connectors. Besides structural elements to allow structuring models in different levels of abstraction, there are elements called Test Patterns that define generic test strategies aiming to test common recurrent behavior on different GUIs.

[1] http://www.m-pathy.com/cms
[2] http://webtango.berkeley.edu
[3] http://www.eyetracking.com/About-Us/What-Is-Eye-Tracking
[4] https://www.crazyegg.com/overview
[5] http://attrakdiff.de/
[6] https://www.optimizely.com/ab-testing/

i. Formal definition of test pattern

A Test Pattern is the set of associated test strategies that may be defined as [2]:
<Goal; V; A; C; P>

- Goal is the ID of the test
- V is a set of pairs variable, value relating test input data with the variables involved in the test
- A is the sequence of actions to perform during test case execution
- C is the set of checks to perform during the test case execution
- P is the precondition (Boolean expression) defining the conditions in which it is possible to run the test

ii. Usability Guidelines

The goal of PBUT is to perform automatic usability tests on web interfaces. So it is necessary to know which common recurrent usability issues can be tested across different web applications.

There are sets of usability guidelines defining the best practices to follow in order to classify a website as a good website from the usability point of view [18]. But, unfortunately, best practices not always are taken into account. These practices may be related to home page usability, task orientation, navigation and IA (Information Architecture), forms and data entry, trust and credibility, writing and content quality, page layout and visual design, search usability and help, feedback and error tolerance. From all these categories, this project focused on usability guidelines related with navigation. Other usability guidelines will be subject of future work.

iii. Reachability Test Pattern

As described previously, there are several usability issue and guidelines that should be followed. One common issue about usability is to assess if it is possible to reach a specific feature within a limited number of steps. So, for this work we selected the guideline described as: "Pages with more importance to the website should be accessed afterwards a few mouse's clicks" practice. This is the pattern built and presented in this paper. We call it "Reachability Test Pattern" and is defined as follows:

- *Goal*: "Is it possible to achieve a feature in n steps?"
- V is the set of input variables and their values, the target URL and the maximum number of steps (n) to perform in order to reach the target URL;
- A is empty. The actions performed aim to explore the GUI under test, are automatic and do not need additional inputs
- C will check if the target URL is achieved in n steps
- P is true. It will always be possible to execute this test strategy

iv. Reachability Test Pattern Execution

The test pattern is executed by exploring the website under analysis and by building a tree where each vertex corresponds to a link (or URL) in the website and each edge corresponds to a possible transition between two links. Thus, based on the depth n, indicated by the user, which corresponds to the maximum number of clicks to reach the destination URL, a tree is constructed resulting from the exploration. Whenever a new URL is reached, it is checked whether it is the destination sought. In constructing the tree we do not consider return transitions and each node is only inserted if it is not already present in the tree. If the desired URL can be reached in n steps, the test passes. If the tree is fully scanned to the indicated depth and the destination is not found, the test fails. In figure 1, you can see an example of a possible tree.

The tree is built using the Breadth-first search algorithm, where all the children of a node are explored before moving to the node below. The sequence of obtaining nodes can be seen in figure 1. Each node has an associated URL achievable through the website exploration, and the children of the current node and their children are always collected until the depth n is reached. The depth corresponds to the maximum number of clicks, n, to reach the destination.

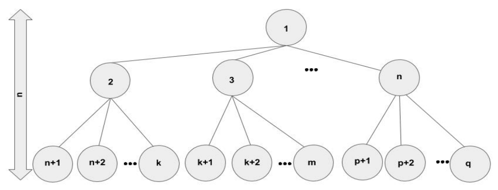

Figure 1: *Representation of navigation tree.*

In the figure 1 the number indicated inside each vertex corresponds to the index of that vertex within the list of vertexes that is stored in memory. It also corresponds to the order in which these nodes are explored. The tree shown is representative of a tree obtained for a depth of 2 clicks.

The value of n is the maximum number of clicks that the user indicated during configuration of the test. All of these nodes correspond to web pages reachable from the homepage. For each current node, only the nodes that can be reached from it and not yet visited can be inserted into the tree. Already visited nodes will not be inserted again in the tree even if they are reached by other elements along the exploration. If an URL external to the website under test is reached during the exploration, it is ignored.

Initially, all the URLs achievable from the homepage are saved in a list of vertexes and the edges saved in a list of edges. These vertexes (1st level of the tree) are then traversed to save information about their children (2nd level of the tree). Afterwards, the nodes of the 2nd level are traversed to save information about their children (3rd level) and so forth. These steps are performed until the maximum depth (n) is reached. For instance, for a depth of 2 clicks, the landing page will have to be found up to the 3rd depth level of the tree.

IV. Case Study

Sigarra ("Information System for Aggregate Management of Resources and Academic Records") benefits from the use of a quality academic information system that promotes the efficiency and effectiveness of its activities, at various levels of administration and management, teaching, research and development and university cooperation activity [19]. Sigarra exists since 1996 (at that time called SiFEUP) and is the result of a consistent strategy and long-term development and organization of an information system capable of meeting the needs of the academic community of a modern university. Sigarra is the subject of our case study, since it is a system used by students, staff and teachers of University of Porto. Just in FEUP (Faculty of Engineering of the University of Porto), Sigarra is used by a universe of about 7500 people. It is a system widely used in the daily life of this number of people, and it is important that certain activities on the website are accessible in an easy and intuitive way. Thus, it is important to test its usability, discovering possible changes that may introduce improvements in its use.

i. Usability Tests on Sigarra

The reachability test pattern allows to analyze the distance of the web pages of a website from its homepage.

We selected two different situations to validate our approach in real scenarios. The first situation is to verify if it is possible to access the study plan of the Master in Software Engineering, from the home page of the site in its English version. The maximum number of clicks expected to access this page is 3 clicks. The test did not pass because it takes more than 3 clicks to reach the goal. This reality makes the task a little more complicated for a visitor aiming to learn more about each course.

This is a usability problem that should be addressed in order to make it easier to obtain the desired information. The click sequence for this test can be seen in figure 2, each arrow corresponds to a click and each figure corresponds to the pages to which the user is redirected after each click. Each figure indicates the area where the click was made with a red rectangle.

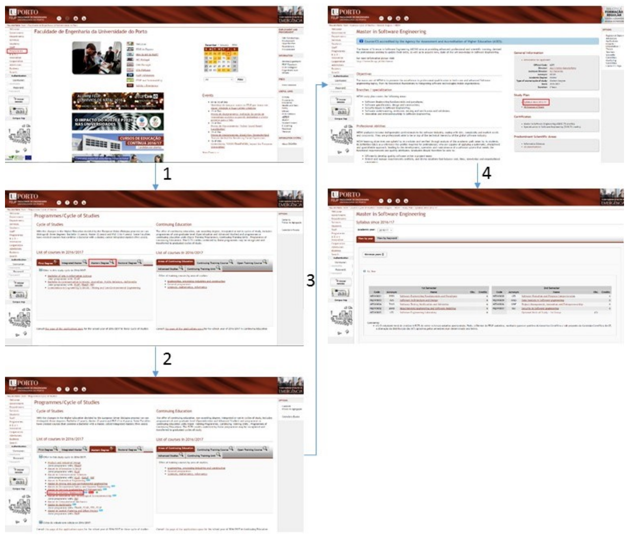
Faculdade de Engenharia da Universidade do Porto
1
Programmes/Cycle of Studies
2
Programmes/Cycle of Studies
3
Master in Software Engineering
4
Master in Software Engineering

Figure 2: *Sequence of navigation to Syllabus of Master in Software Engineering.*

The 2nd situation is to check if it is possible to visualize the academic calendar with a maximum of three mouse clicks. This test passed since it is possible to access that desired landing page with two clicks (see figure 3).

Thus, this case study allowed to assess the effort needed to reach the web pages of a website. It also allows to check if the web pages considered more important (for instance, because they have much requested information or other reason) are reached with less effort. This analysis may be important to increase the usability of the website.

V. Conclusions

This paper presents a new approach to run automated usability testing on web platforms. We defined a test pattern that tests the effort required to access a particular content on a website without user intervention. To do this, the user must indicate the destination URL and the maximum number of steps necessary to reach it. The test automatically scans the application to the indicated depth and indicates whether or not it is possible to achieve that content in that number of steps. The pattern has been validated in an existing application. The result of the study shows that the approach can be applied successfully. There are other recurring usability issues that can be tested in the future, for example, if in every web page of the system it is always possible to return to the previous page without using the back button of the keyboard or browser, among others. In the future, we also intend to improve our approach, including the dynamic behavior of web applications.

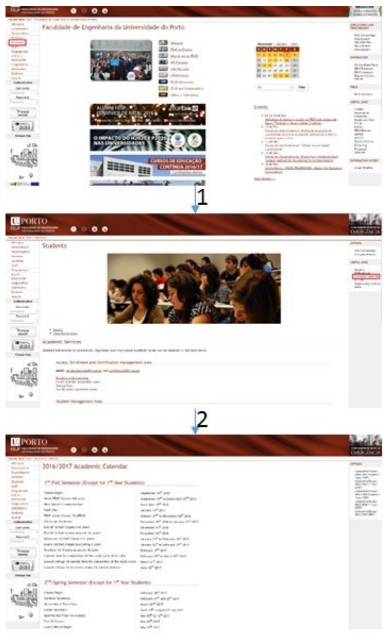
Faculdade de Engenharia da Universidade do Porto
1
Students
2
2016/2017 Academic Calendar

Figure 3: *Sequence of navigation to Academic Calendar.*

References

[1] Rodrigo M. L. Moreira and Ana C. R. Paiva, "PBGT Tool: An Integrated Modeling and Testing Environment for Pattern-Based GUI Testing", 29th IEEE/ACM International Conference on Automated Software Engineering (ASE 2014). September 15 - 19, Västerås, Sweden, 2014

[2] Rodrigo Moreira, Ana C. R. Paiva, and Atif Memon. Pattern-Based GUI Testing. In ISSRE - The 24th IEEE International Sympo-

sium on Software Reliability Engineering, pages 288–297, 2013.

[3] Rodrigo M. L. M. Moreira and Ana C. R. Paiva, "Towards a Pattern Language for Model-Based GUI Testing, Proceedings of the 19th European Conference on Pattern Languages of Programs (EuroPLoP 2014), Kloster Irsee in Bavaria, Germany, 9-13 July, 2014

[4] Rodrigo M. L. M. Moreira, Ana C. R. Paiva, "A GUI Modeling DSL for Pattern-Based GUI Testing - PARADIGM", 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2014), 28-30 April, Lisbon, Portugal

[5] Tiago Monteiro and Ana C. R. Paiva, "Pattern Based GUI Testing Modeling Environment", March 18, Fourth International Workshop on TESTing Techniques & Experimentation Benchmarks for Event-Driven Software - TESTBEDS, Co-located with The Sixth IEEE International Conference on Software Testing Verification and Validation, 2013

[6] Speicher, M., et al. (2014). Ensuring Web Interface Quality through Usability-Based Split Testing. Web Engineering: 14th International Conference, ICWE 2014, Toulouse, France, July 1-4, 2014. Proceedings. S. Casteleyn, G. Rossi and M. Winckler. Cham, Springer International Publishing: 93-110.

[7] International Organization for Standardization (2010). ISO 9241-210: Ergonomics of Human-system Interaction–Part 210: Human-centered Design for Interactive Systems. https://www.iso.org/obp/ui/ #iso:std:iso:9241:-210:ed-1:v1:en *Terms and Definitions*, 2.13.

[8] Atterer, R., Wnuk, M., Schmidt, A.: Knowing the Users Every Move – User Activity Tracking for Website Usability Evaluation and Implicit Interaction. In: Proc. WWW (2006).

[9] m-pathy, //www.m-pathy.com/cms, online on December 2016.

[10] Carta, T., Patern'o, F., de Santana, V.F.: Web Usability Probe: A Tool for Supporting Remote Usability Evaluation of Web Sites. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011, Part IV. LNCS, vol. 6949, pp. 349–357. Springer, Heidelberg (2011).

[11] WebTango: Automating Web Site Evaluation, //webtango.berkeley.edu, online on December 2016.

[12] Eyetracking, //www.eyetracking.com/About-Us/What-Is-Eye-Tracking, online on December 2016.

[13] Crazyegg: //www.crazyegg.com/overview, online on December 2016.

[14] Attrakdiff: //attrakdiff.de/, online on December 2016.

[15] https://www.optimizely.com/ab-testing/, online on December 2016.

[16] Nebeling, M., Speicher, M., Norrie, M.C.: W3Touch: Metrics-based Web Page Adaptation for Touch. In: Proc. CHI (2013).

[17] Rosanna Cassino, Maurizio Tucci, Giuliana Vitiello, Rita Francese Empirical validation of an automatic usability evaluation method, Journal of Visual Languages & Computing, Volume 28, June 2015, Pages 1-22, ISSN 1045-926X.

[18] Usability Guidelines: //webstandards.hhs.gov/guidelines, online on December 2016

[19] Sigarra: //sigarra.up.pt/up/en, online on December 2016