Data-Driven Usability Refactoring: Tools and Challenges

Alejandra Garrido[1], Sergio Firmenich[1], Julián Grigera[2] and Gustavo Rossi[1]
LIFIA. Facultad de Informática. Universidad Nacional de La Plata
[1]Also CONICET [2]Also CIC
La Plata. Argentina
{garrido, sfirmenich, juliang, gustavo}@lifia.info.unlp.edu.ar

***Abstract*—Usability has long been recognized as an important software quality attribute and it has become essential in web application development and maintenance. However, it is still hard to integrate usability evaluation and improvement practices in the software development process. Moreover, these practices are usually unaffordable for small to medium-sized companies. In this position paper we propose an approach and tools to allow the crowd of web users participate in the process of usability evaluation and repair. Since we use the refactoring technique for usability improvement, we introduce the notion of "data-driven refactoring": use data from the mass of users to learn about refactoring opportunities, plus also about refactoring effectiveness. This creates an improvement cycle where some refactorings may be discarded while others introduced, depending on their evaluated success. The paper also discusses some of the challenges that we foresee ahead.**

***Index Terms*—Usability, accessibility, crowdsourcing, refactoring, A/B testing.**

I. Introduction

The number of web applications has grown exponentially in recent years. This growth has led to great advances in technology to present the user with increasingly attractive and interactive solutions. Contradictorily, as the interaction possibilities become more complex, the usability and accessibility of the applications are weakened [1]. Usability problems abound, like pages overloaded with content, confusing processes, inconsistency in the design, or inflexibility in certain operations [2]. In addition, most of today's web applications do not meet the standards of accessibility and are unusable by people with disabilities [3].

Organizations that may afford usability evaluation have usability experts perform mostly guideline reviews and user testing [4]. In user testing, representative users evaluate an application by completing a sequence of typical tasks. The benefit of user testing is that it captures real usage data. The disadvantage is that it is expensive: it requires recruiting test subjects and spending time and resources for experts to design the tests, analyze results, discover problems and find solutions for those problems. While there are tools to automate remote user testing, and crowdsourcing platforms that may be applied for the same purpose (e.g., uTest, test.IO), development teams still need usability experts, at least to analyze the results and find solutions for the problems encountered.

Moreover, learning from the behavior of masses of users and making data-driven decisions is highly valuable [5]. That's the reason why web analytic tools have become so popular to measure traffic, market trends, and system's effectiveness while demanding fewer resources. However, concrete, high-level problems are still hidden behind the statistics, and require a usability expert to uncover them and find the solutions [6]. Meanwhile, A/B testing is usually applied by large organizations to measure market performance of different solutions with statistical significance [5], though the cost of A/B testing may be prohibitive for small companies.

In our previous work we have proposed the use of the refactoring mechanism to improve external qualities of web applications, specifically usability and accessibility, and we have advanced in refactoring tools in the client [2], [7]. One of the benefits of client-side web refactorings (CSWR for short) is that they create transformations by means of scripts, which can be easily instantiated for a particular page and easily installed in the browser. Besides usability and accessibility, CSWR may be used in general to improve user experience (UX) with web applications. The important point is that with this technology, UX improvement is no longer restricted to site owners. While there are approaches like Social Accessibility [8] that recruit volunteer users to improve web accessibility by adding metadata, to the best of our knowledge there is no other proposal to collaboratively solve UX problems, and mainly, no support to evaluate the solutions created by the community.

Consequently, our goal is to allow the crowd of web users to collaboratively participate in the corrective and perfective maintenance of web applications' UX, through a refactoring-and-testing iterative process with three-stages:

1. identification of the UX problems that users suffer, i.e., refactoring opportunities;
2. UX problem repair in terms of CSWR, created by and for the community;
3. evaluation of CSWR through controlled experiments that will ultimately guide the whole process.

In this paper we describe our approach, the tools we have built, and the challenges that remain ahead. We hope to contribute to both: the users' community, to share solutions otherwise unattainable, and the site owners, to learn from the feedback of users about their problems and preferred solutions.

II. Approach and Tools

The enormous number of websites available in the WWW and the amount of different needs and capacities of their users calls for a massive collaborative approach for assessing and improving the UX, that is, a crowdsourcing approach. Crowdsourcing uses the power of the crowd to solve problems, reaching solutions otherwise unattainable or unaffordable. Using an open call for participation, crowdsourcing has been used for some time to solve different types of software engineering problems with high success [9].

Our client-side web refactoring (CSWR) technology makes it possible to apply crowdsourcing at different stages of UX improvement. On the one hand, CSWR allows improving the design of a running application, after learning from feedback of real users about what works and what doesn't [2]. On the other hand, since CSWR are applied on the client, they can be instantiated by volunteers other than website owners, can be distributed to other users, and its effectiveness to solve the problems can be measured again. Thus, the approach lets feedback in the form of web usage data drive the refactoring process: by mining UX problems before and after refactoring we are able assess the real improvement.

We propose a platform for the voluntary, non-anonymous participation of web users in 3 different stages of the process of UX corrective maintenance and improvement: *(A)* UX assessment; *(B)* UX repair and *(C)* repair evaluation.

A. UX Assessment

In the literature of refactoring, a useful concept is that of "bad smells", which relates to the potential problems in the design that may be solved by refactoring. Similarly, we have defined and catalogued *usability smells* as hints to usability problems that may be solved by usability refactorings [2], [6]. The same applies to *accessibility smells* and, in general, we can talk about *UX smells*. Cataloging UX smells allows identifying and classifying problems with the web interface at a higher level of abstraction than that provided by raw statistics. Particularly, we are concerned with UX smells related with the *user interaction* (UI), i.e., patterns of user events that were shown to create problematic interaction.

Thus, regarding the collaboratively assessment and report of UX problems related with the UI, our approach proposes 2 methods: *1)* automatic identification of UX smells from UI events, and *2)* manual report of UX smells.

1) Automatic Detection of UX Smells: we have developed a tool called USF that mines UI events from real users on-the-fly, classifies the relevant ones and analyses them to discover usability smells of user interaction [6]. USF has a client-side component that preprocesses raw UI events and sends relevant ones to the server, where several detection algorithms, optimized through data stream mining techniques [10], allow instant smell reporting without causing a performance downgrade. USF is currently able to detect more than 16 usability smells with good precision. Some examples are: "Undescriptive Element", "Misleading Link", "Free Input for Limited Values" and "Unnecessary Bulk Action".

2) Manual Report of UX Smells: since not all UX smells can be detected automatically, it's important to allow users to make explicit reports of smells. To receive smell reports, we are adapting a tool that gathers web adaptations requirements from final users [11]. With this tool, reporting a UX smell works as "subscribing" to it: in later stages, when solutions to the smell become available, its subscribers will be invited to vote for them (more on this in Section C). While we could use an existing crowdsourcing environment like uTest, test.IO or mTurk to gather UX smell reports, there are some mismatches with our approach. These platforms all propose workers to perform certain tasks, so they are useful to perform remote user testing [12]. We instead are interested to gather problems that occur "in the wild", without predefined tasks. Moreover, we prefer a voluntary participation, where users seek solutions for their UX problems rather than a monetary compensation. This prevents all problems related with fraud [12].

Manual reports of smells should be integrated with automatic ones and clustered similarly until enough instances from different users show their relevance.

B. UX Repair

CSWR implement catalogued solutions for UX smells, by applying transformations to the Document Object Model (DOM) of web pages when installed on the client browser [7]. An example is "Split Page", which solves the problem of a saturated, complex page by dividing it into simpler pages or sections. Each CSWR is coded as a generic configurable script that applies a well-known solution or design pattern, and gets instantiated by providing specific parameters like the URL (or URL pattern) of the target webpage and the xPaths of the DOM elements where the changes are applied.

We have constructed a repository of CSWR in their generic form (called *abstract CSWR*), and have evaluated them in two aspects: (*i*) their effectiveness in improving *usability in use* of e-commerce applications [13] and the *accessibility* of websites like Facebook, LinkedIn, MercadoLibre and Gmail [7], and (*ii*) the implementation effort perceived by developers [13].

Our goal is to involve the crowd of web users to participate in instantiating CSWR to solve the UX smells captured automatically or manually by other users. For this purpose, our crowdsourcing platform will work as a *UX smell tracking system*. To promote a massive use of our system, we've developed two strategies to simplify CSWR instantiation for non-programmers: *1)* automatic, and *2)* visual instantiation.

1) Automatic Instantiation of CSWR: we have developed a tool called Kobold that is fed with bad smell reports from USF, suggests the appropriate CSWR to solve the smell, and in some cases it may instantiate the CSWR automatically [14]. Some refactorings that Kobold may create automatically are: "Add Autocomplete", "Add Validation", and "Add Processing Page". Thus, given a UX smell, a volunteer would request the system to suggest an appropriate CSWR, and in many cases, instantiate it automatically. In other cases Kobold requests specific parameters from users for a semi-automatic instantiation.

2) Visual Instantiation of CSWR: we've built a tool that allows users point-and-click on the target page to select the components that will act as values for each parameter [7]. Our next step is to combine this tool with Kobold's semi-automated instantiation, so appropriate CSWR parameters may be visually selected.

Once CSWR are instantiated either automatically or visually, they are submitted to the platform for the next stage. Note that with this strategy, we are able to control the kind of scripts fed in the system to be only known CSWR instances created with our tools.

C. Repair Evaluation

Having a UX smell tracking system makes it possible to evaluate the effectiveness of each refactoring in solving smells, and use the results to retrofit the process of continuous improvement. For this purpose, we propose a strategy similar to A/B testing, which splits the universe of users to compare alternative solutions, under the premise that most of the ideas fail and it's essential to experiment frequently [5]. With this strategy, when multiple CSWR are proposed to solve a given smell, all of them are laid to compete with each other and with the original version (until a significant number of results are gathered). Note that our "universe of users" includes all registered members of the platform that visit the page under test, including the subscribers of the smell being solved.

In the case of traditional A/B testing, metrics reflecting revenue or costs are defined to decide the winning version [5]. In our context, we don't have such a metric to compare versions. The simplest approach is to execute the evaluation stage similarly to the UX assessment stage (*A*): collect UX smells both automatically and manually, though filtering those occurring at the same DOM element(s) of the original smell. When the evaluation stage is over, the deciding criterion is: (*i*) discard those CSWR that collected the same or any smell; (*ii*) keep those CSWR that did not collect any smells. To decide among the remaining refactorings, we propose a voting mechanism among those users that subscribed to the original smell (those that reported the smell manually in stage A).

Although we are building automated support in our platform for this stage, including random assignment of users to test cells and verification of ending conditions, a person is needed to decide when to start the A/B test and with which CSWR. This job is reserved to trusted community members, so we propose using a *trustworthiness indicator* based on their involvement and performance, as proposed elsewhere [15].

III. Challenges

There are several interesting challenges that may be discussed in the context of the proposed approach.

A. Challenges in UX Assessment

1) Manual Smell Reports from Disabled Users: with respect to the accessibility requirements, an important challenge that we have is to adapt the tool, nowadays mostly visual, to be easily accessed by users through a screen reader.

B. Challenges in UX Repair

1) Resilience to DOM Update: CSWR are instantiated with a URL pattern and the xPath of applicable DOM elements. Thus, instantiated refactorings depend heavily on the DOM, and may break upon a page update. We propose a strategy that would detect these updates automatically and feed them in the platform as a special kind of smell, inviting members to repair the parameters of the CSWR instance under the new DOM.

2) Managing Dependences between Refactorings: CSWR depend on others that modify the same DOM element. In previous works we defined some guidelines to compose refactorings according to the impact of changes [7]. However, composing instantiated CSWR automatically while guaranteeing the consistence of the composition is still a challenge. A strategy we intend to explore is to introduce some concepts from concurrent computing, and see DOM elements as shared resources that a CSWR may lock to prevent its changes to be spoiled.

3) Encouraging Adoption: motivation is an important antecedent for obtaining contributions in crowdsourcing communities. Previous work shows that there exist a clear intention of web users to improve their UX voluntarily [11]; thus, they are motivated to participate. However, if the cognitive effort for making a contribution is inadequate, volunteers may be discouraged. To encourage adoption, we aim at reducing the effort required for every step in our approach, and at recognizing user participation by merit (increasing their *trustworthiness*).

4) Preventing Design Corruption: allowing the crowd to alter the website design may be considered harmful, as it could lead to break its consistency. For example, Yale's art school allows users to alter its website, which produces some chaos (art.yale.edu). This challenge may be solved in two ways: (*i*) let trusted members review submitted solutions and start the evaluation stage only with selected CSWR which appear safe and useful; (*ii*) after the evaluation period, let the platform members decide if they want to uninstall a solution.

C. Challenges in Repair Evaluation

1) Other Ways to Measure CSWR Performance: as we mentioned before, applying A/B testing for usability with no information of the task being executed, makes it very difficult to find appropriate metrics to decide the winning CSWR (among all those proposed to solve the same smell). Apart from our current strategy, we plan to explore a different approach in the future: using the UI event log in USF to infer the task being executed at the time of collecting the smell (through sequential pattern mining), and measure *usability in use* of the different versions [13].

2) Identifying Malicious Scripts: our current approach is to allow members to instantiate existing CSWR only through our tools. This approach may be considered too restrictive, especially since we'd like to allow new abstract CSWR to be defined. However, allowing new JS scripts to be added poses a

security risk for users. We foresee two ways to tackle this problem, which could be even combined. First, we could let members with a higher trustworthiness indicator be in charge of evaluating new abstract CSWR before they enter stage C. Secondly, we could define a Domain Specific Language (DSL) for creating new abstract CSWR; this DSL should allow defining the kind of changes over the DOM elements that CSWR require, without using JS. In this way, we would prevent to execute any malicious JS sentence.

3) Hierarchical Community vs. Automatization: having higher ranked members in crowdsourcing communities makes it easier to control certain tasks; in our system, trusted members have the power to start the evaluation stage, and discard unsafe or unuseful solutions. However, this may cause a task overload on a small subgroup of members, known as *herding* [16]. In such case, we could resort to automate the start of the evaluation stage. Moreover, to avoid frustrating members with bad solutions during the whole evaluation period, we may let them uninstall unwanted CSWR at any time, which would count as a negative vote.

IV. Related Work

There are several tools aimed at improving usability and accessibility through a crowdsourcing approach. Takagi et al. proposed the Social Accessibility service to receive problem reports from end users and collaboratively add accessibility metadata to webpages [8]. These authors have also explored the challenges of sustainable crowdsourcing services to support accessibility, and conclude that the main success factors for this kind of approaches are the simplicity of the tasks, their management, and the decisive role of top contributors. A similar platform is Social4All, which allows to collaboratively create accessible profiles for different websites [3].

Regarding usability evaluation, CrowdStudy is a tool that incorporates crowdsourcing techniques to recruit volunteers to perform remote user testing [17]. It differs from our approach since it requires usability experts to design user tests with guided tasks and associated usability metrics. Similarly, uTest is a crowdsourcing service provider specific for usability testing [12]. Users must follow test scripts composed of several tasks, upload screenshots and keep track of timing. Instead, our approach doesn't require experts, it gathers UX smells in the wild, and it may even go unnoticed to users.

V. Conclusion

Usability, accessibility and UX in general are very important aspects of the software engineering process of a web application. However, outside the HCI community, these aspects do not receive enough attention. Most importantly, there are not enough tools to make UX maintenance and improvement affordable for small/medium companies, mainly to learn from feedback of their users and try different ideas. We believe that a collaborative UX maintenance process involving the community of users may give an answer, both to the companies that want to improve the UX of their applications and to the users for which their problems usually go unheard.

Finally, our approach provides an environment for research on globalization issues of web applications, making it possible to wisely mine UX smells from crowds of different locations and cultures, which may drive localized refactorings.

Acknowledgment

This research is supported by ANPCyT-Argentina grant PICT-2015-3000.

References

[1] J. Nielsen and H. Loranger, *Prioritizing Web Usability*. Pearson Education, 2006.

[2] A. Garrido, G. Rossi, and D. Distante, "Refactoring for Usability in Web Applications.," *IEEE Softw.*, vol. 28, no. 3, pp. 60–67, 2011.

[3] R. Gonzalez Crespo, J. Pascual Espada, and D. Burgos, "Social4all: Definition of specific adaptations in Web applications to improve accessibility," *Comput. Stand. Interfaces*, vol. 48, pp. 1–9, 2016.

[4] A. Fernandez, E. Insfran, and S. Abrahão, "Usability evaluation methods for the web: A systematic mapping study," *Inf. Softw. Technol.*, vol. 53, no. 8, pp. 789–817, 2011.

[5] R. Kohavi and R. Longbotham, "Online Controlled Experiments and A/B Tests Motivation and Background," *Encycl. Mach. Learn. Data Min.*, pp. 1–11, 2015.

[6] J. Grigera, A. Garrido, J. M. Rivero, and G. Rossi, "Automatic detection of usability smells in web applications," *Int. J. Hum. Comput. Stud.*, vol. 97, pp. 129–148, 2017.

[7] A. Garrido, S. Firmenich, G. Rossi, J. Grigera, N. Medina, and I. Harari, "Personalized web accessibility using client-side refactoring," *IEEE Internet Comp*, vol. 17, no. 4, pp. 58–66, 2013.

[8] H. Takagi, S. Harada, D. Sato, and C. Asakawa, "Lessons Learned from Crowd Accessibility Services," in *Int. Conf. Human-Comp. Interaction. LNCS-8117*, 2013, pp. 587–604.

[9] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *J. Syst. Softw.*, 2016.

[10] L. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2014.

[11] D. Firmenich, S. Firmenich, J. M. Rivero, L. Antonelli, and G. Rossi, "CrowdMock: an approach for defining and evolving web augmentation requirements," *Requir. Eng.*, pp. 1–29, 2016.

[12] D. Liu, R. G. Bias, M. Lease, and R. Kuipers, "Crowdsourcing for usability testing," in *Proc. ASIST*, 2012, vol. 49, no. 1.

[13] J. Grigera, A. Garrido, I.Panach, D.Distante,G.Rossi, "Assessing refactorings for usability in e-commerce applications," *Empir. Softw. Eng.*, vol. 21, no. 3, pp. 1224–1271, 2016.

[14] J. Grigera, A. Garrido, and G. Rossi, "Kobold: Web Usability as a Service," in *ASE 2017 - Tool Demonstrations*, 2017, p. to appear.

[15] Y. H. Tung and S. S. Tseng, "A novel approach to collaborative testing in a crowdsourcing environment," *J. Syst. Softw.*, vol. 86, no. 8, pp. 2143–2153, 2013.

[16] H. Yu, C. Miao, C. Leung, Y. Chen, S. Fauvel, and V. R. Lesser, "Mitigating Herding in Hierarchical Crowdsourcing Networks," *Sci. Rep.*, vol. 6, no. 4, 2016.

[17] M. Nebeling, M. Speicher, and M. Norrie, "CrowdStudy: General toolkit for crowdsourced evaluation of web interfaces," in *ACM Symp. Eng. Interac. Comp. Sys.*, 2013, pp. 255–264.