Assessing Refactorings for Usability in E-Commerce Applications

Julián Grigera[1], Alejandra Garrido[1*], Jose Ignacio Panach[2], Damiano Distante[3] and Gustavo Rossi[1*]

[1]*LIFIA, Fac. de Informática, Univ. Nac. de La Plata, Argentina*
[julian.grigera, garrido, gustavo]@lifia.info.unlp.edu.ar

[2]*Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica, Universitat de València, Valencia, España*
joigpana@uv.es

[3]*Unitelma Sapienza University, Rome, Italy*
damiano.distante@unitelma.it

[*] *Also CONICET, Argentina*

ABSTRACT

Refactoring has been reported as a helpful technique to systematically improve non-functional attributes of software. This paper evaluates the relevance of refactoring for improving usability on web applications. We conducted an experiment with two replications at different locations, with subjects of different profiles. Objects chosen for the experiment were two e-commerce applications that exhibit common business processes in today's web usage. Through the experiment we found that half of the studied refactorings cause a significant improvement in usability. The rest of the refactorings required a post-hoc analysis in which we considered aspects like user expertise in the interaction with web applications or type of application. We conclude that, when improving quality in use, the success of the refactoring process depends on several factors, including the type of software system, context and users. We have analyzed all these aspects, which developers must consider for a better decision support at the time of prioritizing improvements and outweighing effort.

Keywords: refactoring; quality in use; usability measurement; web engineering; software maintenance and evolution

1. INTRODUCTION

Refactoring is a fundamental process to revert the decay of a software system by applying changes that do not alter the observable behavior of the system (Fowler & Beck 1999). Refactoring is applied over working software (even small pieces) that can be observed in action, measured, and tested. The premise is to find opportunities for improvement of non-functional attributes, while preserving the functional ones. When properly addressed, refactoring fits in the software lifecycle as part of each development iteration, by first identifying and analyzing design problems (called "bad smells" in the refactoring literature), and then selecting the appropriate refactorings to solve them.

Refactoring was initially defined to improve the internal quality of code and turn it maintainable and extensible, but it was later extended to include the improvement of external quality attributes, such as performance (Rieger et al. 2007; Dig 2011), security (Maruyama 2007) and usability (Garrido et al. 2011). Regarding usability, refactoring can be a helpful technique to iteratively and systematically improve the way users interact with a graphical user interface (GUI) when performing common tasks. In a previous work, we defined a catalogue of refactorings to improve usability, in the specific field of web applications and their business processes (Distante et al. 2014). Refactorings listed on that catalogue aim at correcting problems on web applications regarding complex navigation, obstructive interaction flow, or awkward content presentation, with the objective of making tasks easier to perform, but preserving the task essence; i.e. its input and results – just like source code refactoring preserves behavior. An example is the refactoring *"Make explicit the steps composing a process and the current step being executed"*.

Martin Fowler did a good job at cataloguing bad smells of poor code quality and linking each one to a set of refactorings that may be used to fix it, depending on the context and the developers' choice (Fowler & Beck 1999). However, it is still hard for developers to make the right decision when they face alternative refactorings to solve a given bad smell. Usability refactorings have their own bad smells, which are also described in the catalogue. They can be discovered through different symptoms like users' complains, usability testing, or customers/revenue drops in the case of commercial websites. The special advantage of web applications (as opposed to desktop/mobile software) is that they are easier for developers to monitor in real time, and get quick feedback from a (potentially) large mass of visitors. The challenge is to be able to gather all this feedback, filter it and rate the remaining usability problems considering their relevance in the business and the consequent importance of solving each one. Although there is a vast amount of research in the area of usability, principally guided by Nielsen (Nielsen 1999; Nielsen & Tahir 2002) and Shneiderman (Shneiderman & Plaisant 2005), developers do not have any systematic technique to discover bad usability smells and prioritize them. Moreover, once developers find a bad smell, they might have to choose among different refactorings to solve it, which is not always an easy decision - just as it happens with source code refactoring.

This article shows the results of an empirical evaluation of how end-users perceive the impact of usability refactorings on web applications. The experiment design is based on the ISO/IEC 25010 definition of *usability*: *the degree to which specified users can achieve specified goals with effectiveness in use, efficiency in use and satisfaction in use in a specified context of use* (ISO 2011). Thus, we have measured effectiveness in use, efficiency in use, and satisfaction in use over the refactored and non-refactored versions of two web applications: an online store and an auctions website. The evaluation has been performed with students in two replications. We have also surveyed the developers who applied the refactorings on the code of both applications to measure the perceived difficulty of implementing each refactoring.

The contribution of this work is two-fold. First, it provides the first empirical evidence on the real effect of usability refactorings for end-users of web applications. Second, it adds the cost of implementation effort to weight and prioritize refactorings in terms of both effect and cost, with the purpose of helping developers choose among many alternative refactorings, or the best sequence of application to maximize the return. We believe that applying the most valuable refactorings first enhances the refactoring process towards improving the business in the least amount of time, with a positive impact on the overall cost of the project. Moreover, prioritizing refactorings involves an optimization of the software development process since we can discard refactorings that are difficult to implement and whose effect in usability is weak.

This paper is organized as follows. Section 2 presents related work on the evaluation of other types of refactorings, interaction patterns, and usability evaluation. Section 3 describes our approach of refactoring to improve usability and lists the refactorings that we evaluated. Section 4 presents details of the experiment definition and planning. Section 5 discusses threats to validity of the experiment. Section 6 explains all the analysis performed on evaluated subjects, and finally Section 7 presents conclusions and future work.

2. RELATED WORK

This section reviews different assessments in refactoring and usability improvement, but as many of the authors claim, the empirical results published in both fields are scarce. However, it is essential for developers to know the effect that any improvement method has in terms of software quality.

In the field of business processes, Fernández-Ropero et al. (Fernández-Ropero et al. 2012) present a quality assessment study that measures the impact of business process refactorings in order to suggest the most appropriate ones according to different scenarios. Note that, as opposed to the refactorings in our work, those refactorings are internal to process models, and have the intent of improving the understandability and maintainability of business process models. The study is very thorough, but has no statistical background to confirm the improvements that refactoring provides. An interesting point of this work is the assertion that developers should not apply refactoring indiscriminately; they must evaluate the cost and benefits depending on the case. The work of Zibran et al. (Zibran & Roy 2011) makes the same conclusion after performing a thorough analysis on effort and benefits of applying source code refactoring, mostly related to code cloning. They also propose an approach for automatically scheduling the refactoring task, by estimating effort and risk for each refactoring. This sort of cost/benefit analysis is also a topic we address in our work by including an effort metric for each refactoring.

The work of Zou et al. (Zou et al. 2007) also evaluates improvements on business processes, but in their case, measuring the impact on usability of e-commerce applications. The enhancements the authors propose, however, are not refactorings, but the introduction of contextual help for end-users through long or difficult tasks. To assess the improvements in different usability aspects like efficiency or satisfaction, they provide a sound statistical test, although the users sample is small: 12 subjects total, 2 expert users of the application under test, and 10 novice users. For the experiment, all users perform actions in both versions of the same application, suffering the learning effect threat to validity in the case of novice users, especially considering that learnability was one of their response variables. Authors however quantify this effect with a second test and claim that it is not significant.

Barnes et al. (Barnes & Vidgen 2003) present an evaluation similar to the one reported in the present work. With the aim of assessing the quality of an e-government site for aspects like usability, aesthetics and navigation, they evaluate quality metrics before and after a redesign process. Using the WebQual tool (Barnes & Vidgen 2000), authors remotely gather subjective ratings from end-users across different locations. The study gathered results from 65 subjects in the first phase and 59 subjects after the site’s redesign, and measured variables related to usability, design and information quality (e.g., ease of use, navigation, appearance or relevance). The result consists of a series of metrics for average rankings and deviations, but it does not include a statistical test. Moreover, their experiment has a different focus than ours, since it evaluates the quality impact of the overall redesign work, while we aim at evaluating the impact of each usability improvement separately.

In the literature of code refactoring, there are also some empirical evaluations worth citing about the influence of refactoring during the development process. Kim et al. performed a quantitative study of the benefits and challenges of refactoring at Microsoft, through surveys with 328 developers and analysis of version history data (Kim et al. 2012). The survey found that developers are not tight to rigorous behavior-preserving transformations while refactoring, which implies higher cost and risks. The analysis of Windows 7 version history confirmed code quality improvements after refactoring, involving reduction of inter-module dependencies and post-release defects. Murphy-Hill et al. accomplished most likely the largest experiment of software refactoring, spanning 13,000 developers and 2500 development hours (Murphy-Hill et al. 2012). Their intent is not to measure quality improvement but to study how developers approach the refactoring process. Their findings confirm that programmers apply refactoring frequently, interspersing them with other program changes, that the refactorings are mostly mid-to-low level and performed manually. The purpose of their experiment, similarly to the work of Vakilian et al. (Vakilian et al. 2012), is to influence the design of future refactoring tools.

In a related field, patterns in web application design (Van Duyne et al. 2007) provide well-known principles for a better graphical interface and interaction design, which impact on usability and other external quality aspects. Dearden et al. present a comprehensive review of interaction pattern languages (Dearden & Finlay 2006) assessing the value of using pattern languages in the field of design in Human-Computer Interaction (HCI). In their work, the authors observe that there is scarce concrete evaluation of such patterns in the literature, and we have not found any statistical evaluation whatsoever. Early work on refactoring for web usability was devised as refactoring-to-patterns aimed at reaching interaction patterns' implementations, like "Introduce Information on Demand" (Garrido et al. 2009). Other web refactorings are targeted to improve specific usability factors (Garrido et al. 2011) like *navigability* (quality of the navigation structure in facilitating organized and effortless access to the application's contents through links), *credibility* (the application's capability to encourage trust) or related aspect like *accessibility* (degree to which a web application can be used by people with physical impairments). Examples of these refactorings are “Convert images to text” for accessibility (Harold 2008), "Replace unsafe GET with POST" for credibility (Harold 2008) and "Turn attribute into link" for navigability (Garrido et al. 2011).

The work from Olsina et al. proposed applying WebQEM, a framework for web quality measurement and evaluation, to identify needs for improvement, recommend web refactorings and assess their impact on specific quality attributes like *learnability*, *operability* and *information suitability* (Olsina et al. 2008). The approach is described through a case study, but there is no statistical experiment to demonstrate the real improvement gain that web refactorings produce on usability, like the one presented here.

Another interesting work to consider is the review of usability evaluation methods presented by Fernandez et al. (Fernandez et al. 2011). Authors point out that 90% of the studies are conducted at implementation stages, when it is already late to change requirements. Refactoring is well suited for adjusting already deployed applications, so it can help developers at late development stages.

3. REFACTORING FOR WEB USABILITY

Fowler defined refactoring as a change that improves software quality, and as the improvement process itself (Fowler & Beck 1999). He catalogued changes that improve non-functional qualities of software like source code's readability, understandability, maintainability, and extensibility. After Fowler's book, many other works have extended the refactoring catalog to other languages, to other software artifacts, and with new intents that go beyond internal quality. For example, there are refactorings for HTML code like "*Enable Caching*" (Harold 2008), and refactorings over AJAX applications that migrate XML to JSON (Ying & Miller 2013), both with the intent of improving performance.

In previous works, we have proposed refactorings for web applications at both model and code level, with the intention of improving usability and accessibility (Garrido et al. 2011; Garrido et al. 2013). We have also catalogued several refactorings specifically designed to improve business processes of e-commerce websites (Distante et al. 2014). One of the refactorings in our catalog is *"Keep the user up to date on the ongoing process"*, aimed at incrementing users' trust and satisfaction while executing a process, by adding information on the state of the process. For instance, this refactoring may be applied to a shopping process by displaying extra data on a shopping cart icon, like content and total amount, so users can always see what they are buying while browsing products in the catalogue. At the model level, this refactoring affects the navigation nodes involved in the process, extending them with new data attributes, and the presentation model for those nodes, since their graphical interface needs to accommodate the new information. Therefore, the target artifacts of web applications' refactorings are specific to the navigation, graphical interface and interaction design, and the intent is quality in use improvement. More precisely, a *usability refactoring* is a change over the navigation or presentation of a web application that is perceived by the final user. It is intended to improve the application's usability, and it preserves the set of use cases and requirements that the application satisfies, and can be checked through acceptance tests. In other words, behavior preservation of a usability refactoring means that it does not modify any business rule, nor it adds any new system-driven behavior (a behavior unrelated to the user interface and its enhancement), neither it breaks any user acceptance test that applies to the web application (Distante et al. 2014).

The objective of this work is to measure through an experiment the real impact of usability refactorings on final users. Table 1 shows our working set of 21 refactorings, with a short description of their purpose. In the rest of the article, when we mention *refactorings* in general, we will be referring to this specific working set, and will identify particular refactorings with the number in the leftmost column of Table 1. Appendix A provides precise details of the exact place where each refactoring was applied in the case studies of the experiment.

4. EXPERIMENT DEFINITION AND PLANNING

We conducted our experiment with two replications, one in Argentina (Ar replication) and the other in Spain (Sp replication). The following is the experiment definition following the template of Basili (Basili et al. 1994):

> The *goal* of this experimental investigation is to compare quality in use, specifically usability, **before** and **after** applying the refactorings *with the purpose of* identifying the effects that each refactoring has on usability as perceived by end-users. The *focus* is set on the differences that end-users experience with and without refactorings during their interaction with the system. The *perspective* is of researchers and practitioners interested to investigate the effect of refactorings for end-users.

In order to fulfill the goal of the experiment, i.e., evaluate usability before and after applying refactorings, we needed different web applications in two versions: original (or non-refactored) and refactored. We recruited 3 students that played the role of developers to implement the refactorings and to take some measurements in the process. We will detail this procedure later in section 6.3.

Table 1. Evaluated usability refactorings

Refactoring ID	Name	Description
R1	Improve the description of process links	Change the text of a link that starts a process to describe it more precisely
R2	Split page	Divide a cluttered page in two or more pages or page sections
R3	Anticipate a validation activity	Avoid revisiting forms to correct data by splitting the whole form validation so that each field is validated right after it is filled
R4	Change widget	Replace the widget used to execute a process activity with a more clear and/or easier to use one
R5	Add a verification activity	Introduce an activity intended to verify for example that a request to a website originates from humans (such as CAPTCHA tests)
R6	Add a "confirm and commit" activity	Introduce a "confirm and commit" activity as the last step before completing a process and committing the associated transaction to increase system trustability
R7	Aggregate activities	Join two simple activities in one page to shorten a process (avoiding unnecessary navigation)
R8	Introduce information on demand	When there is plenty of information to show in a small area, use the same screen space to show different parts of the content according to the user's choice (by hovering or clicking an active object)
R9	Turn attribute into link	When there are page contents which clearly refer to other contents (pages) such as product names, book authors, etc., add a link to be able to navigate to the related content
R10	Provide breadcrumbs	Add a visual aid to help users keep track of their navigation path up to the current page
R11	Move widget	Move a presentation/interaction widget from one page to another where it is more significant
R12	Split list	Similarly to 'Split Page', divide a list of items that became too long, into more, easier to access lists
R13	Distribute menu	Distribute a menu of actions affecting a list of elements to each element individually
R14	Keep the user up to date on the ongoing process	While executing a process divided in several pages, add information to each page about the current step
R15	Improve link destination announcement	Enrich the anchor of a link to better communicate its target page or process and avoid the user to erroneously click on them
R16	Remove duplicated process links	When a process link that leads the user to a given action is duplicated on a page, remove the duplication to improve consistency
R17	Make explicit the steps composing a process and the current step being executed	When executing a process in several steps or pages, add information to each step about the whole set of activities composing the process, current step and estimated time to complete
R18	Enable user to go back in the process steps	When executing a step in a process, add a back button to allow the user to revisit and possibly change a previous step
R19	Add a summary activity	As part of a long process, add an activity to enable the user verifying the current status of the process before committing
R20	Add processing page	When a long transaction must take place before giving a result to the user, add a widget or intermediate page that displays feedback about the status or progress of the transaction
R21	Add an "assistance" activity	Introduce a system activity such as autocomplete to help the user accomplishing a task

4.1 Subjects and Objects

We recruited different *subjects* for both Sp and Ar replications to play the role of final users of web applications. The choice of different locations allowed us to compare two different profiles of users: software developers (Ar subjects) and regular end-users (Sp subjects). The choice is based on the idea that software developers working as end-users could take better advantage of the GUI improvements than other (non-developers) end-users. We aimed to contrast whether the usability improvement perceived per refactored activity depends on the users' profile.

In the Sp replication the volunteers were 22 students of the Degree in Information and Documentation of the University of Valencia (Spain). These subjects plan to be future librarians and their interaction with computers throughout the day is not frequent. In the Ar replication we recruited 27 subjects working at the LIFIA research center of the University of La Plata (Argentina); 22 of them were undergraduate students, and the remaining 5 were PhD students, all of them in the field of Computer Science, and most of them working daily in the development of software systems. Given their background, we expected Ar subjects to use web applications more frequently than Sp subjects, which could be used in favor of having a broader representative population for the experiment. Note, however, that even though Ar subjects are software developers, they are not necessarily experts in the interaction with systems similar to the experimental objects, and as such, we also had some novice users in the Ar group (see Table 3).

Table 2 shows a summary of the subjects' experience in the labour market. Note that the Sp replication is mainly composed of subjects with jobs unrelated to Computer Science. Conversely, subjects of the Ar replication commonly interact with web applications, since most of their jobs are related to Computer Science.

Table 2. Subjects' work experience

	Sp Replication	Ar Replication
Students only	12	3
Jobs NOT related to computer science	10	1
Jobs related to computer science	0	23
Total	**22**	**27**

We defined two problems as *objects* for the experimental investigation: an online store and an auction website. In both cases, we installed popular pre-made frameworks (Zencart for online stores and WeBid for auctions) in a controlled environment. Applications are similar in that both allow end-users to register and browse through products organized in categories. Users may also acquire products and edit their account data. The main difference between both applications lies in how customers acquire the products: in the auction website they place bids and then pay for auctions if they win, while in the online store they buy products directly through a traditional shopping cart and checkout process. Each subject worked with both problems in such a way that a particular subject never used the same object twice

Table 3 classifies the subjects depending on their previous experience with the two types of objects used in our experiment. Again, note that Ar subjects use online stores and auctions' sites more frequently than Sp subjects. Moreover, in both cases, there are more subjects that use online stores than subjects using auctions websites. The content of both Table 2 and Table 3 was obtained through a demographic questionnaire that was filled in before running the experiment.

Table 3. Subjects' experience in the use of web applications

		Very Frequently	Usually	Temporarily	Sporadically	Never	Total
Sp	**Frequency of use of online stores**	1	2	7	5	7	22
	Frequency of use of auctions' sites	0	1	1	4	16	22
Ar	**Frequency of use of online stores**	3	6	10	7	1	27
	Frequency of use of auctions' sites	1	2	8	9	7	27

In order to measure the effect on usability of each particular refactoring, we had to ensure that all subjects faced the effects that each refactoring produced on a GUI. To do this, we planned a set of tasks (one for each problem/website) that subjects had to complete, e.g. "register to the website". These tasks represent typical use cases for end-users in e-commerce/auction websites. Each task was in turn composed by smaller-grained activities that subjects had to follow. This way, the activities that compose each task exercised at least one refactoring (in the refactored version) and thus worked as measurable entities from which we extracted information about the impact of refactorings in web GUIs. After selecting the tasks, we made sure they covered all the refactorings for both problems. A comprehensive list of all tasks and their activities appears in Appendix A.

4.2 Research Questions and Hypothesis Formulation

The goal of the experiment is to study the effect that each refactoring has in *quality in use*, specifically *usability in use*. Since ISO/IEC 25010 defines *usability in use* as a software quality whose measures are *effectiveness in use*, *efficiency in use* and *satisfaction in use*, our research questions are defined using these three measures of usability, for each particular refactoring:

- RQ1: *Is effectiveness in use affected by refactoring?* This research question aims at determining whether or not each particular refactoring improves effectiveness in use. ISO/IEC 25010 defines effectiveness in use as *the degree to which specified users can achieve specified goals with accuracy and completeness in a specified context of use* (ISO 2011). According to this definition, to answer the research question for each refactoring we must measure the percentage of success obtained per completed task (accuracy and completeness). The null hypotheses to address this research question is:

  ***$H_0$1**: The effectiveness in use of a system after refactoring is similar to the effectiveness in use of a system before refactoring.*

- RQ2: *Is efficiency in use affected by refactoring?* This research question aims to determine whether or not a particular refactoring improves efficiency in use. ISO/IEC 25010 defines efficiency in use as *the degree to which specified users expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.* According to this definition, to answer the research question for each refactoring, we must measure the end-users' time to perform the activities involved in the refactoring, considering the reached level of effectiveness. The null hypotheses being tested to address this research question is:

  ***$H_0$2:** The efficiency in use of a system after refactoring is similar to the efficiency in use of a system before refactoring.*

- RQ3: *Is satisfaction in use affected by refactoring?* This research question aims to determine whether or not a particular refactorings improves satisfaction in use. ISO/IEC 25010 defines satisfaction in use as *the degree to which users are satisfied in a specified context of use.* According to this definition, to answer the research question for each refactoring, we must measure end-users' personal opinion about their satisfaction while performing the activities on the refactored applications. The null hypotheses being tested to address this research question is:

  ***$H_0$3:** The satisfaction in use of a system after refactoring is similar to the satisfaction in use of a system before refactoring.*

4.3 Factors, Response Variables, and Metrics

Factors are the exploratory variables that an experiment studies to measure their effect on the response (Juristo & Moreno 2010). The factor of this experiment is the use of the refactoring process. This factor has two levels or treatments: to apply or not the refactoring process. For each problem (i.e. *object*), we created two versions of the same web application: *refactored* and *non-refactored*. The refactoring process referred to in this article applies changes to web applications in their navigation, presentation, and business process design aspects. Regarding navigation, refactorings aim to optimize the way users access the website's sections and the available navigation structure. Regarding presentation, refactorings improve the graphical interface by removing unnecessary elements or replacing widgets for more appropriate ones. Regarding business processes, refactorings improve their design to support complex actions (like publishing a new product for auctioning) without imposing a burden on users for unnecessary long, confusing or duplicated activities.

Response Variables are the values measured in the experiment to study how the factors influence them (Juristo & Moreno 2010). The response variable to answer *RQ1* is *effectiveness in use*, whose metric (*M1*) is the level of success (correct completion) reached during the execution of an activity. M1 can be seen like a black box with two possible values for each activity: 1, which means that the activity was finished successfully; 0, which means that the activity failed. Effectiveness values for refactorings involved in more than one activity were averaged. For example, if refactoring RX was involved in 4 activities, 3 successfully completed and 1 unsuccessful, the averaged value would be 0.75. This naturally results in values between 0 and 1 (including decimals) to measure the effectiveness in use of refactorings shared with different activities.

The variable to answer *RQ2* is *efficiency in use,* whose metric (*M2*) is the percentage of effectiveness divided by the time spent in the activity. For each activity execution, we measured the completion time in seconds. Then we calculated the effectiveness-time ratio. If a refactoring affects multiple activities, the efficiency values for each activity were aggregated through the quotient of the effectiveness average by the time average (i.e. an average divided by an average). The possible values for efficiency are positive numbers.

The variable to answer *RQ3* is *satisfaction in use*, whose metric (*M3*) is a 5-point Likert scale captured with a questionnaire. The possible answers for each statement in the questionnaire are: *'Totally agree'*, *'Fairly agree'*, *'Undecided'*, *'Fairly disagree'* and *'Totally disagree'*. At the end of each task, we displayed satisfaction statements to evaluate the refactoring effect as perceived by the user. Generally, these likert-scaled statements stated assertions on the activities' ease of use, or general satisfaction. For instance, after Task 3.2 on the Zencart website (requesting a product search), the assertion that pops up is: "It was easy to search for a specific product in the catalog of the store". Likewise, at the end of Task 4.4 on the WeBid website (asking to find the status on a recently placed bid), the assertion states "It was easy to check the status of my bid after I did it". Each refactoring was evaluated by at least one satisfaction question, and some refactorings required more than one question to get the accurate subjects' opinion; for example, the refactoring "Anticipate a validation activity" was used multiple times for different kinds of input, like dates or credit cards. Refactorings used in several activities and refactorings with more than one satisfaction question were aggregated through the average. It should be noticed that a Likert scale is ordinal and the use of average is not suitable for ordinal values. However, since our scale's values can be deemed equidistant, we think it does not cause a large distortion to treat them as quantitative values and it does cause a benefit for the purpose of the experiment to aggregate them. Note that this idea is not new, several authors have worked with data extracted from Likert questionnaires as quantitative data. For example, Blaikie (Blaikie 2003) and Cohen et al. (Cohen et al. 2007) also assume that Likert-type categories constitute interval-level measurements, which allows to calculate averages as an aggregation technique. There are other authors such as Moody et al. or Jönsson and Wohlin that have applied arithmetical operation to data extracted from a Likert questionnaire (Moody et al. 2003; Jönsson & Wohlin 2004). Other authors such as Bruun et al. or Wnuk et al. use also the average to aggregate items of a Likert-scale (Bruun et al. 2014; Wnuk et al. 2013). Therefore, the possible values for satisfaction in use in our experiment are numbers between 1 and 5.

4.4 Instrumentation

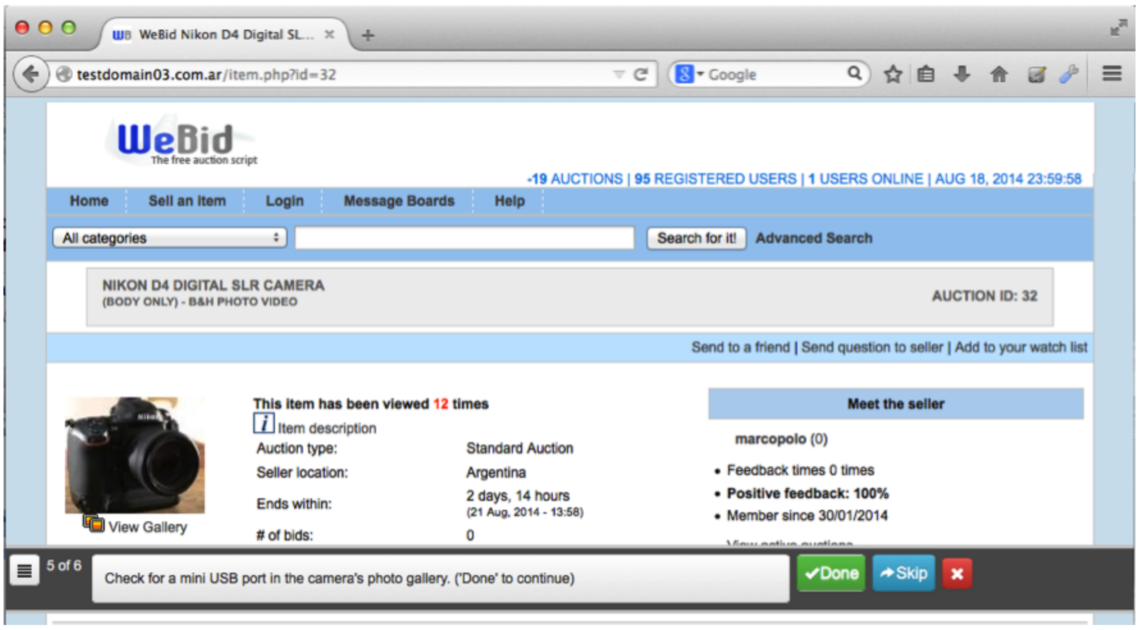
To carry out the experiment, we created a tool that guides users through tasks and activities (Guide Tool). The tool helps to determine whether or not the activity was finished successfully (used for metric M1), measures the time spent in each activity (used for M2), and presents satisfaction questionnaires when required (used for M3), saving subjects' answers. Using the Guide Tool simplified the mechanics of the experiments considerably, guiding the subjects through the activities and gathering all the measurements automatically.

The Guide Tool was devised as an Add-On for the Mozilla Firefox web browser. Subjects were able to easily install this Add-on in their browsers, and from that point on, load the different sets of activities from a menu. Fig 1 shows the tool in action, as a bar on the bottom of the screen. On the left of the bar there is a button for users to see all the activities for the current task, in the middle are the instructions for the current activity, and on the right there are the control buttons "Done" (to manually indicate completion of the current activity), "Skip" (to skip to the next activity), and "Cancel" (to quit the task).

At the beginning of the experiment, the tool displays a demographic questionnaire to gather the background of the subject, their experience in interacting with web applications, and their knowledge of software development. Next, the tool starts guiding subjects through each task and activity, and when necessary provides extra data (e.g., a test credit card number).

The Guide Tool automates the measure of M1 as much as possible. It is able to check whether a navigation-based activity is successful through different techniques, like checking destination URL patterns, or asserting the presence of specific GUI elements. However these techniques are not enough to verify the successful completion of all the activities (M1). In order to solve this problem, the tool requests extra information from the subject to determine the success or failure of some specific activities. For example, when the activity required the user to search for a specific product, the Guide Tool asked for the product's price. In another example, the Guide Tool asked for the total cost of the shopping cart to verify the completion of the purchase task. For the activities in which questions did not make sense (6 out of 27 in Zencart and 5 out of 33 in Webid), we simply verified if users had skipped the activity, causing the tool to log it as unsuccessful. For example, in Zencart's task 3.4 that asks the subject to read a review, there is no easy way of automatically asserting whether the review was indeed read, so subjects can choose between the "Done" and "Skip" buttons to indicate whether they finished reading or they just skip the task (either because they did not find the review or they just did not want to read). We allowed skipping single activities to prevent users from aborting a full task just because of a single unfinished activity. At the end of each task the tool sends all the data to a server, along with the user's related information.

When a subject finishes an activity, the tool moves on to the next one and saves the time spent to finish the activity (M2). Most of the time, the tool is able to detect when an activity is finished, otherwise it asks the user (see Fig 1). The Guide Tool automatically performs the measure for M2 for all the activities.


WeBid Nikon D4 Digital SL...

testdomain03.com.ar/item.php?id=32

WeBid The free auction script

-19 AUCTIONS | 95 REGISTERED USERS | 1 USERS ONLINE | AUG 18, 2014 23:59:58

Home | Sell an item | Login | Message Boards | Help

All categories | Search for it! | Advanced Search

NIKON D4 DIGITAL SLR CAMERA
(BODY ONLY) - B&H PHOTO VIDEO

AUCTION ID: 32

Send to a friend | Send question to seller | Add to your watch list

This item has been viewed 12 times
Item description
Auction type: Standard Auction
Seller location: Argentina
Ends within: 2 days, 14 hours (21 Aug, 2014 - 13:58)
# of bids: 0

View Gallery

Meet the seller
marcopolo (0)
- Feedback times 0 times
- Positive feedback: 100%
- Member since 30/01/2014

5 of 6
Check for a mini USB port in the camera's photo gallery. ('Done' to continue)
Done | Skip

Fig 1 Tool for gathering usability data running on sample auctions site

We have also defined a satisfaction questionnaire using a 5-point Likert scale to extract subjects' opinion about the use of refactorings (M3). To gather this input, the tool presented a questionnaire after the subject had completed a group of activities (e.g., after filling up a registration form, which encompasses 3-4 activities), since showing questions after each individual activity was too burdensome. Each refactoring has at least one question to ask subjects about their satisfaction during the activity execution. Questions ask about the aspects of usability that each refactoring is supposed to improve. We use the same questions for both versions of problems (with and without refactorings). We have defined our own satisfaction questionnaire since existing ones are too generic and are not powerful enough to extract all the characteristics related to refactorings.

4.5 Experiment Design

In this section we describe the design choices we made to limit threats to validity. The design is a *paired design blocked by experimental objects* (Juristo & Moreno 2010), since we have two values for each response variable per subject (see Table 4). The paired design allows us to work with the largest sample size, since we avoid dividing subjects per treatment in order to contrast each other. Another benefit of applying both treatments to all the subjects is that variability among subjects is avoided. We have used two problems P1 and P2 (i.e., two web applications based on Zencart and WeBid) in such a way that every subject interacted with both of them. This blocks the possible effect of the problem in the results, reducing the dependence on problem threat. Each problem was implemented with two versions (the two levels of the factor): one version included refactorings (R) and the other version did not (¬R).

In order to avoid the learning effect threat, each subject interacted with only one version of each problem. This way each treatment was applied through only one version of only one problem. This results in four possible combinations for subjects to perform the experiment, depending on the order of interaction with each version of the problem. Table 4 details each combination. We have balanced the subjects assigned to each group in order to block the effect of starting with a specific problem or treatment. It should be noticed that the 4 combinations took place at both locations. We recruited the following subjects per combination and location: #1, 6 in SP and 6 in AR; #2, 6 in SP and 7 in AR; #3, 5 in SP and 7 in AR; #4, 5 in SP and 7 in AR. In order to avoid the threat of dependences between the order of interaction with problems and results, we balanced the four possible combinations among all subjects randomly. We provide a detailed discussion on these and other validity threats in Section 5.

Table 4. Experiment design

Combinations	Zencart (P1)		Webid (P2)	
	R	¬R	R	¬R
# 1	1st			2nd
# 2		1st	2nd	
# 3		2nd	1st	
# 4	2nd			1st

4.6 Experiment Procedure

Figure 2 shows a graphical summary of the procedure for our experiment, according to the experimental design. Before the experiment, we classified each subject into one of four possible combinations of our design (see Table 4). Groups were assembled to be as balanced as possible for each combination.

The experiment procedure starts when the subject fills in the demographic questionnaire. Next, depending on the combination assigned, the subject starts the interaction with one of the web applications (P1 or P2), with or without refactorings. Then, for each activity to complete, the subject reads the activity's instructions and tries to perform the activity on the web application. The tool saves, in a transparent way, the percentage of completeness of the activity and the time spent in the execution. This data is used for metrics M1 and M2, to calculate effectiveness and efficiency respectively. Once a subject finishes a group of activities, a satisfaction questionnaire is presented to calculate metric M3. The process is repeated for all remaining activities. As described in Section 4.4, subjects may skip an activity that they do not know how to complete, and in that case, the correspondent metrics are discarded.

Note that all activities and questionnaires are equivalent in both versions of the same problem. This way we can contrast effectiveness, efficiency and satisfaction with the same metrics for both versions through the same instruments.

Once a subject finished interacting with the first web application, the experiment continues with the other problem. This second problem consists of a different web application using a different treatment. This assignment depends on the combination initially assigned to the subject (Table 4). The process in this second round is exactly the same we used in the first one. At the end of the procedure, for each subject, we have data for a web application with refactorings and data for the other web application without refactorings.

5. THREATS TO VALIDITY

This section discusses the threats to the validity of our experiment according to Wohlin et al. (Wohlin et al. 2012) and how we have tried to avoid or minimize their effect.

Conclusion validity: this validity is concerned with issues that affect the ability to draw the correct conclusion about relations between the treatment and the outcome. We have considered the following threats: *Low statistical power,* which appears when we work with less than 30 subjects. In our experiment we have 22 and 27 subjects in each replication respectively. We have minimized this threat using a statistical test powerful enough for such amount of subjects: Wilcoxon test for repeated measures, which is non-parametric. The use of repeated measures is justified because treatments are applied to the same subject twice. These subjects are our experimental units since we observe the results of applying treatments through them. The possibility to skip activities results in a few empty metrics, which decreases the number of data to analyze. The experiment suffers from this threat only in those refactorings with empty metrics for some subjects. We have minimized its effect by discarding subjects with more than 10% of skipped activities. All the discards have been applied to the Sp replication, which was composed of 36 subjects initially. *Reliability of measures,* which appears when we cannot trust in applied measures. We have minimized this threat automating as many measures as possible. Metric of efficiency and metric of effectiveness (when possible) are calculated through the Guide Tool automatically, reducing human mistakes. Metric of satisfaction is the only one that suffers this threat, since we cannot ensure that some subjects made a mistake when they filled in the satisfaction questionnaire. The satisfaction questionnaire used has been defined ad-hoc for this experiment, since existing ones are too generic. This results in a lack of validation of our questionnaire, since we are the only ones that have used it for the moment.

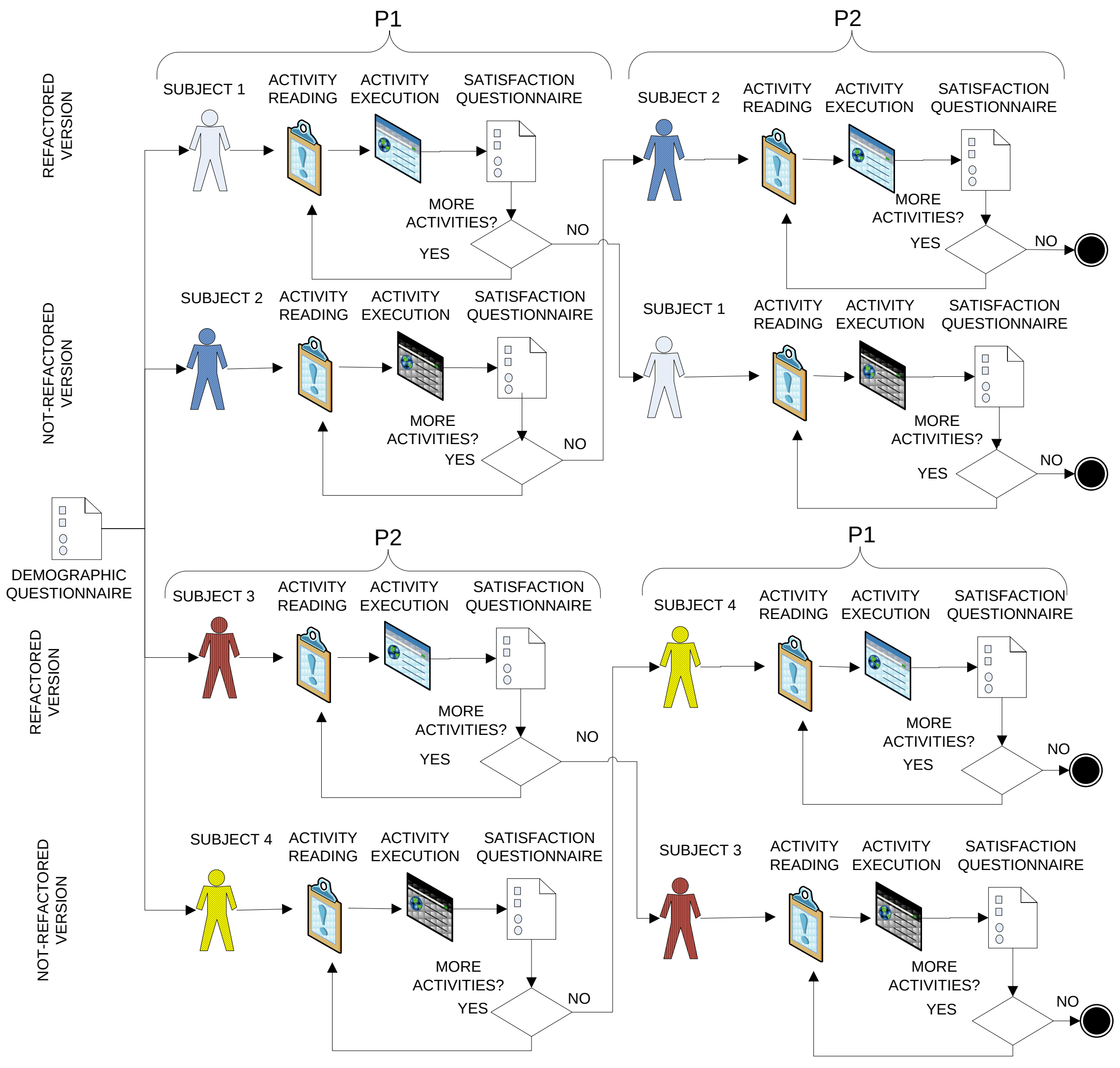


Fig 2 Graphical summary of the experiment procedure

Additionally, some refactorings were applied in more than one activity. To process their data, we have aggregated all the metrics by refactoring through the average. During the aggregation, we might have lost some significant results. Therefore, metrics for effectiveness, efficiency and satisfaction suffer this threat for refactorings that appear in more than one activity (R2, R3, R4, R7, R11 in Zencart and R1, R2, R3, R4, R9, R15, R16 in Webid). *Reliability of treatment implementation*, which means that the treatment is implemented in a different way for different subjects. We have avoided this threat since we force subjects to interact with the same problems under the same conditions. *Random irrelevancies in experimental settings*, which means that elements outside the experiment may disturb the results. For example, interruptions due to mobile calls or chats with other subjects can result in a time extension that affects results regarding efficiency. We have minimized these external effects monitoring subjects in a room. *Random heterogeneity of subjects*, which happens when the group of subjects is too heterogeneous. There is a risk that the variation due to individual differences is larger than due to the treatment. We have avoided this threat by recruiting subjects of similar profiles in each replication.

Internal validity: this validity concerns influences that can affect the factor with regard to causality, without the experimenter's knowledge. We have considered the following threats: *History*, which means that the circumstances at different times may not be the same and may affect the results. We have avoided this threat running the experiment for all the subjects at the same time. In order to avoid that the order of applying each treatment affects the results, we have balanced the number of subjects that start the experiment with each treatment. *Maturation*, which means that subjects react differently as time passes. We have avoided this threat limiting the experiment to 2 hours, which was a time more than enough to finish the tasks. Subjects who spent more than 2 hours are not considered in the analysis. Moreover, subjects could skip the tasks that they did not know how to complete. Subjects that skipped more than

10% of activities were not considered in the analysis in order to avoid many metrics without value. *Instrumentation,* which means that a bad design in the instruments definition may result in erroneous results. We have avoided this threat checking all our instruments through a pilot test with two subjects before running the experiment. *Selection,* which means that results obtained with a reduced number of volunteers are not representative of the whole population. We have minimized this threat replicating the experiment with subjects of two different profiles: subjects that develop web applications and subjects that interact with web applications as end-users. *Resentful demoralization,* which appears when there are subjects that receive less desirable treatments. We have avoided this threat applying both treatments to all the subjects.

Construction validity: this validity concerns generalizing the results of the experiment to the theory behind the experiment. We have considered the following threats: *Restricted generalizability,* which means that results cannot be generalized to any context. We have minimized this threat considering two problems and replicating the experiment in two groups. The use of two problems helps disengage results from a specific problem and the replication using two groups disengage results from a specific profile of subject. In spite of our effort to minimize this threat, we cannot ensure that our results can be generalized for any problem or type of user. *Hypothesis guessing,* which means that subjects may guess the hypothesis behind the experiment and base their behavior to satisfy them. We have avoided this threat hiding the treatment that each subject received in each problem. This way, subjects cannot benefit one treatment unconscientiously since they did not know if they were interacting with the refactored version or with the non-refactored version. *Evaluation apprehension,* which appears when subjects are afraid to be evaluated. We have minimized this threat recruiting only volunteers.

External validity: this validity concerns the conditions that limit our ability to generalize the results of our experiment to industrial practice. We have considered the following threats: *Interaction of selection and treatment,* which appears when we have a subject population that is not representative of the population we want to generalize. We have minimized this threat using different profiles of subjects in each replication. The Ar replication is based on software developers, while the Sp replication has typical end-users that have never developed software systems. Moreover, we have different degrees of expertise of users in Sp replication: those that use web applications frequently and those that interact with web applications sporadically. This allows generalizing the results to all these different types of end-users: software developers, experts in the use of web applications and novice users of web applications. *Interaction of history and treatment,* which appears when the time in which each treatment is applied may affect the results. We have avoided this threat applying both treatments at the same time. *Interaction of setting and treatment,* which appears when the experimental setting or material is not representative of reality. We have minimized this threat using two web applications similar to real web applications of online stores and auctions. Note that we could not use real web applications since we had to modify their code to include refactorings.

6. ANALYSIS AND INTERPRETATION

We divide the analysis and interpretation into four subsections. The first one describes the analysis obtained from the statistical test at first hand. The second part shows details of a post-hoc analysis where we analyzed characteristics such as users' expertise, type of web application, and alternative treatment for null values obtained when subjects skipped activities. In order to weight the cost-benefit ratio for each refactoring, the third subsection presents an analysis of the implementation effort of the refactorings. This effort is gathered together with the preliminary and post-hoc analysis in the fourth subsection, to create a prioritized list for the use of refactorings. This last part also presents a discussion with insights of the whole experiment and a summary of results. Table B.1 in Appendix B shows descriptive data for each response variable, including the number of valid observations, the minimum, the maximum, the mean and the median for each variable per refactoring.

6.1 Preliminary Analysis

Since we have a design within-subjects that does not follow a normal distribution (K-S test obtains a *p-value* higher than 0.05), we applied Wilcoxon test for paired samples with $\alpha = 0.05$. Remember from Section 4.2 that our null hypotheses are: ***$H_0$1***: "The effectiveness in use of a system after refactoring is similar to the effectiveness in use of a system before refactoring"; ***$H_0$2***: "The efficiency in use of a system after refactoring is similar to the efficiency in use of a system before refactoring"; ***H03***: "The satisfaction in use of a system after refactoring is similar to the satisfaction in use of a system before refactoring". Table 5 shows our results to test the null hypotheses for each refactoring. A number "1" in the table means that the refactored version of the web application got significantly better results that allow rejecting the null hypothesis in favor of the refactored version. Meanwhile, a number "-1" means that the non-refactored version was better, i.e., the null hypothesis was rejected in favor of the non-refactored version. Finally, a "0" means that there were no significant differences to reject the corresponding null hypothesis.

Table 5. Experiment results. Results that favor refactoring are greyed out

Refactoring ID	Sp Replication			Ar Replication		
	Effectiveness	Efficiency	Satisfaction	Effectiveness	Efficiency	Satisfaction
R1	0	0	0	1	1	1
R2	0	0	0	0	0	0
R3	-1	0	0	0	1	1
R4	0	0	0	0	-1	0
R5	0	0	0	0	0	0
R6	0	0	0	0	0	1
R7	0	-1	0	0	-1	0
R8	0	0	0	0	1	0
R9	0	0	0	0	1	0
R10	0	1	1	0	1	1
R11	0	0	0	0	0	0
R12	0	0	0	0	0	0
R13	0	0	0	0	0	0
R14	0	0	1	0	0	1
R15	0	0	0	0	0	1
R16	0	0	0	0	0	0
R17	1	0	1	0	0	0
R18	0	0	1	1	1	1
R19	0	0	0	0	-1	0
R20	0	0	1	0	0	0
R21	0	-1	0	0	0	0

The values in Table 5 let us derive some insights:

- There are 11 refactorings that were statistically proven as beneficial in some aspects of 'usability in use' in at least one replication of the experiment.
- The refactoring process does not show significant improvement in all cases, and further analysis is required in each situation.
- There are more significant refactorings in the Ar replication than in the Sp replication, possibly related to the difference in user's profiles.
- There are no refactorings that display negative results in satisfaction. This allows us to conclude that user experience is enhanced or at least preserved by refactoring.

The rest of this section categorizes the refactorings in three major groups: *beneficial refactorings* (11), *doubtful (or with unproven benefit) refactorings* (6), and *unbeneficial refactorings* (5). It should be noticed that these counts add up to 22 refactorings, when we only have 21; this happens because R3 belongs in two categories at once: *beneficial* and *unbeneficial,* since it favors both refactored and non-refactored versions in different aspects of each replication. Beneficial refactorings are further classified by the replication in which they impacted (both locations, Ar and Sp). For each refactoring we describe how it was applied to the problems, what activities were involved, α values, and the Cliff's delta non-parametric effect size. The α value (*p-value*) shows whether there is a significant difference between treatments of each factor (refactoring versus non-refactoring). When α is higher to 0.05 we cannot reject the null hypothesis, which means that there are not significant differences between both treatments. The effect size indicates the magnitude of the difference between both treatments (Grissom & Kim 2005). We use this technique since our response variables do not follow a normal distribution. Cliff's delta (*d)* effect size ranges in the interval [−1, 1] and is considered small for 0.148 ≤ *d* < 0.33, medium for 0.33 ≤ *d* < 0.474, and large for *d* ≥ 0.474. A positive sign means that values of the treatment with a refactored version are greater than the values of the non-refactored version (inversely for a negative sign).

Moreover, for each group of *p-values* we have applied the Benjamini-Hochberg (Benjamini & Hochberg 1995) technique to avoid false significances. Since we performed a large number of statistical tests, some might have *p-values* less than 0.05 purely by chance, even if all our null hypotheses are really true. A *p-value* of 0.05 means there is a 5% chance of getting our observed result. For this reason, we have performed a false discovery rate control to correct our multiple comparisons. After applying the Benjamini-Hochberg technique to *p-values* of Tables 6, 7, 8, 9, 10 and 11, we conclude that there are not false significant discoveries.

6.1.1 Beneficial refactorings

The direct results of the statistical test show that 11 of the 21 tested refactorings provide a **clear benefit**, that is, their results in the analysis allowed to reject at least one of the three null hypotheses in favor of the refactored version of the applications. Thus, we considered them important for improving usability and should have a high-priority in the refactoring process of a web application.

There are two possible justifications for the differences in significance between the Ar and Sp replications. One justification is that software developers (Ar subjects) can take better advantage of the usability improvements as compared with non-developers (Sp subjects), which was our presumption as stated in Section 4.1. Evidence shows that user's profile did have an influence in perceived usability. The second justification for the differences between both locations is the statistical power, which is related to the number of subjects. There were 22 subjects in Sp and 27 in Ar. These numbers may result in a low statistical power. The power of any statistical test is defined as the probability of rejecting a false null hypothesis. Low values of power mean that non-significant results may involve accepting null hypotheses when they are false. According to G*Power (Faul et al. 2007), an α of 0.05 and an effect size of 0.4 (for example), need a sample size of 73 to ensure that null hypothesis cannot be rejected with absolute certainty. This means that with our sample size we do not reach the highest statistical power. Therefore, with a bigger sample size, more null hypotheses could be rejected, which may result in less difference between Ar and Sp replications. With our data, it is difficult to know if divergences between Ar and Sp replications are due to users' profiles or statistical power.

Next, we analyze beneficial refactorings divided in three groups: *(i)* refactorings significantly beneficial in both replications, Sp and Ar, *(ii)* refactorings significantly beneficial in Ar, and *(iii)* refactorings significantly beneficial in Sp.

(i) Refactorings significantly good for Sp and Ar replications

The refactorings that proved significant benefits in both replications of the experiment are R10, R14 and R18. Table 6 lists the refactorings with their α and effect sizes in order of decreasing significance. It displays with grey background significant results. The following paragraphs explain the way we applied refactorings in Table 6 to each activity and discuss their results.

Table 6. *P-values* and effect size for refactorings beneficial at both locations

	Sp						Ar					
Ref. ID	Effectiveness		Efficiency		Satisfaction		Effectiveness		Efficiency		Satisfaction	
	α	Effect size	α	Effect size	α	Effect size	α	Effect size	α	Effect size	α	Effect size
R10	1	0.0004	0.01	0.5	0.05	0.41	0.157	0.009	0.001	0.52	0.007	0.54
R18	0.926	0.009	0.08	0.36	0.03	0.44	0.01	0.59	0.01	0.49	0.01	0.45
R14	0.317	-0.11	0.4	-0.17	0.04	0.35	0.317	-0.004	0.884	-0.009	0.009	0.53

R10) *Provide breadcrumbs*: the intent of this refactoring is to help users keep track of their navigation path up to the current page, allowing them to quickly navigate back to one of the previous navigation steps. We use this refactoring in both problems (the online store and the online auction site) to improve navigation in the hierarchical structure of product (or bid) categories. With breadcrumbs, it gets easier to search and browse for other products in the same or related category. This is the recommended use of breadcrumbs (Nielsen 1999). When there are no breadcrumbs, customers must use the search tool or restart navigating the categories from the root each time, which is expensive and cumbersome (Van Duyne et al. 2007). Table 6 shows that this refactoring proved to significantly improve efficiency and satisfaction in both replications. Moreover, satisfaction was even more significant in Ar than in Sp, including a larger effect size.

R18) *Enable user to go back in the process steps*: this refactoring allows users to return to a previous step during a process. Without this refactoring, a process only allows forward navigation, forcing users to start over if they wish to correct data on a previous step. Note that this refactoring is similar to the previous one in the sense that it makes navigation more flexible, providing "Multiple Ways to Navigate" (Van Duyne et al. 2007). The absence of a mechanism to navigate back in a process forces users to depend on the browser's back button, which does not ensure maintaining the current state of the process. In case the current state is not saved, users need to start the process all over again.

In the problems of the experiment, this refactoring was applied to allow navigating back to change data in the purchase process (online store) and in the selling process (auction site). In the Ar replication, the refactoring showed a significant improvement in all three usability aspects. In the Sp replication, the improvement was slighter, but it did improve the level of satisfaction. A possible reason for R18 not showing significance in efficiency or effectiveness in Sp, is that subjects of this replication were more used to the browser's back button. Since tasks are the same for both R and ¬R versions, we cannot ensure that subjects that interacted with the R version did not use the back button instead of the navigation provided by this refactoring.

R14) *Keep the user informed on the ongoing process:* this refactoring adds information to display the current state of a process, like the contents of the shopping cart and the total price, or the state of a bid. Without this refactoring, to get the information a user needs to navigate to the shopping cart in the online store, or to the product and its current bids in the auction site, which is burdensome and may divert the user of the ongoing process. That is why we believe this refactoring showed a large significance for satisfaction in both replications. However, the use of R14 did not reduce the time to finish the task or the number of mistakes. A possible reason to justify the absence of significance for effectiveness and efficiency is that most subjects did not read carefully all the extra information provided by this refactoring. Moreover, the activity where R14 was measured involved two other refactorings, and the aggregation of results may have influenced it negatively.

(ii) Refactorings significantly beneficial in the Ar replication

The refactorings with significant results in Ar only are R1, R3, R6, R8, R9, and R15. They are listed in Table 7 in order of significance (the higher in the table the more significant). Significant values are represented on grey background while non-significant are on white background. Next paragraphs describe these results in detail.

Table 7. Alpha values and effect sizes for refactorings beneficial in the Ar replica

Ref. ID	Effectiveness		Efficiency		Satisfaction	
	α	Effect size	α	Effect size	α	Effect size
R1	0.04	0.15	0.002	0.72	0.001	0.45
R3	0.763	0.008	0.006	0.32	0.005	0.42
R15	0.317	0.002	0.265	0.3	0.002	0.55
R6	0.166	0.22	0.332	0.25	0.01	0.31
R8	1	0.009	0.03	0.2	0.382	0.007
R9	0.157	0.002	0.05	0.2	0.463	-0.001

R1) *Improve the description of process links*: this refactoring makes links easier to understand by altering their text or adding icons. Like it happens in many websites, the ¬R versions on both problems had a "Login" link, including in the target page the registration form. Subjects were required to register and in both websites it was hard to find the registration form. Contrarily, the R version of both problems had a "Register" link apart from the "Login" one. In the ¬R version, subjects usually got lost before registering, and in many cases reported that they were expecting to find a registration link below the login form, an often-repeated pattern. In the Ar replication, this refactoring showed benefits for all the three variables of usability in use.

R3) *Anticipate a validation activity*: this refactoring adds inline validation to a given form, preventing users from submitting incomplete or invalid data. The rationale behind the satisfaction improvements of R3 may be that correcting mistakes on the spot is usually less annoying than doing it after submission, since the latter requires re-locating the missing or wrong fields. Efficiency boost is reasonable, since inline validation usually leads to a single successful submission. Effectiveness did not improve significantly in the R version with respect to the ¬R version, but this also seems logical if we consider that the main task affected by R3 is **registration**. Since registration is a mandatory step to continue browsing the site as a logged user, the subjects in the ¬R version might have felt the need to overcome any problems in order to continue with the test in either version, instead of abandoning it in such an early stage; thus, we expected high effectiveness in the ¬R version just as much as in the R version.

R15) *Improve link destination announcement*: this refactoring introduces the pattern "Link Destination Announcement" (Nanard et al. 1998), which purpose is to enrich the anchor of a link with additional widgets to present extra data of the target page. This is helpful to prevent unnecessary navigations. We use it for instance in the online store to add to the shopping cart icon the number of items and total cost of the cart. In the auction site, the ¬R version has the currency of the bid price as a link to the currency converter. In the R version, we added the image of a calculator besides the bid price to highlight the link and better describe its target. Although the gain in satisfaction was expected in the R version, efficiency and effectiveness did not improve significantly. We did however observe efficiency improvements of 24% average on the raw numbers.

R6) *Add a "confirm and commit" activity*: this refactoring adds a summary at the end of a task with multiple steps, providing users with a chance to go through the choices they made before submitting. This kind of refactorings will hardly bring any benefits in efficiency, since they *add* an extra step that will entail more time to complete the task. Instead, we expected an improvement over satisfaction, which was actually the case. Effectiveness, on the other hand, is not significantly affected, which is not surprising given that having a summary is not expected to alter the task's success.

R8) *Introduce information on demand*: since this refactoring allows obtaining information by hovering but without the need for navigation, efficiency improved most likely due to the savings in time. However, it may be harder for novice users to find this feature, and that is probably why there was no significance in the Sp replication. In the online store application, the task improved by this refactoring was that of checking on a product's ratings: in the ¬R this meant scrolling to the end of the product's page (hence, extra time), while the R version showed a pop-up with the required information when hovering over the rating's stars. The counterpart activity on the auctions' website was similar, but related to the *other auctions* of the user.

R9) *Turn attribute into link*: similarly to *Introduce information on demand*, this refactoring has shown significant improvements in efficiency for the R version, most likely due to the time savings in searching and navigation. In this case, the subjects on the online store site were asked to read the reviews with the specific grade of 2 stars, which in the R version was directly accessible by clicking on the "2 stars" icon, while the ¬R version required a manual search. The auctions' site had a similar activity but about the "Terms and Conditions" of the site. Again, the feature introduced by this refactoring may go unnoticed for novice users, which would explain the lack of significance in Sp.

(iii) Refactorings significantly beneficial in the Sp replication.

The refactorings with significant results in Sp are R17 and R20, and their α and effect sizes appear in Table 8. Significant values are represented on grey background while non-significant ones are on white background.

Table 8. Alpha values and effect for refactorings beneficial at SP

	SP					
Ref. ID	**Effectiveness**		**Efficiency**		**Satisfaction**	
	α	**Effect size**	α	**Effect size**	α	**Effect size**
R17	0,05	0,26	0.408	0.35	0,02	0,28
R20	1	0.002	0.215	-0.2	0,05	0,3

R17) *Make explicit the steps composing the process and the current step being executed*: this refactoring is used during the long checkout process on the online store site, and the publishing process on the auctions site. A significant effectiveness gain on the R versions means that users were able to complete the task, while in contrast, users of the ¬R version were unable to complete the process. This is a very important result, even if efficiency is not significantly improved. The gain in satisfaction can be explained by the harsh reviews on frustrated users of the ¬R applications that could not complete the tasks.

R20) *Add a processing page:* this refactoring informs the progress of a long process being executed. There were no efficiency boosts, which we actually expected since the time the process takes is actually the same in both R and ¬R versions; the only difference is in how the system informs the user what is happening in the background. Although there was no significant effectiveness improvement, this could have happened if more users had quit the task in the ¬R version, not knowing whether there was a failure of some nature or the process was just taking too long. Satisfaction did show a significant improvement in this case.

6.1.2 Doubtful Refactorings

There are six refactorings in Table 5 for which we could not get enough evidence to reject the null hypothesis. They are: R2, R5, R11, R12, R13, and R16. In next section, we apply a post-hoc analysis to study what is happening with them for this lack of significance. We focus our analysis on the following issues: user experience, type of activity/system, or by treating missing data differently.

6.1.3 Unbeneficial Refactorings

In Table 5 there are five refactorings that show negative results. This means that the ¬R version obtained better measures than the R version for at least one aspect. There is only one refactoring which turned out inefficient in both replications: R7 (*Aggregate activities*). Besides, there are two refactorings that resulted inefficient only in Ar: R4 (*Change widget*) and R19 (*Add a summary activity*) and one more that resulted inefficient in Sp: R21 (*Add an assistance activity*). Only refactoring R3 (*Anticipate a validation activity*) showed very good results in Ar in terms of efficiency and satisfaction, but resulted ineffective in Sp. Next, the post-hoc analysis section discusses the possible reason why these refactorings significantly decrease usability in use instead of improving it.

6.2 Post-hoc Analysis

There are ten refactorings in our working set whose effect in web applications seems to be not significant or counterproductive (i.e., that do not show benefit in any aspect). That is, 6 refactorings did not produce statistical data to reject any of the null hypotheses (R2, R5, R11, R12, R13, R16), and 4 of them did not show any benefit and in some aspect rejected the null hypotheses in favor of the ¬R version (R4, R7, R19, R21). Moreover, refactoring R3 shows benefits in the Ar replication but proved unbeneficial in Sp. In order to study the reasons for these outcomes, we performed a post-hoc analysis. We first analyzed two features that can affect the results of the experiment: the experience of users, and the type of web application. Finally, we also studied an alternative for processing null values to be able to incorporate in the analysis of paired samples those subjects who skipped some activities. The next three subsections describe the details of these new analyses.

6.2.1 Experience of users

There are different profiles of subjects in each replication; some subjects frequently use web applications that are similar to those used in the experiment, while others have not previously interacted with any of them. This subsection analyzes the possibility of obtaining significant refactorings only for one of these profiles of subjects, since some refactorings might be more suitable for expert users and others for novices. We considered as *novices* the subjects that *never* used systems such as an e-store or e-auction (code 5 in the demographic questionnaire, see Table 3) or *sporadically* use them (code 4). All other subjects, who used at least one of both systems, (codes 1 to 3 in either system) were considered *experts*. Thus, in the Sp replication we have 10 experts and 12 novices, and in Ar replication we have 19 experts and 8 novices.

Table 9 displays all results considering user experience, where M1=*Effectiveness in use*, M2=*Efficiency in use* and M3=*Satisfaction in use*. Similarly to Table 5, a value of 1 means a significant improvement using the refactoring, -1 means a significant worsening using the refactoring, and 0 means the absence of significant differences.

In general, Table 9 shows that the same values of the preliminary analysis are preserved for each row, although we can see how each value is singled out for a particular experience level. For example, in the case of R3, Table 5 had a “-1” in effectiveness in the Sp replication, and Table 9 shows that the value occurs only for novices (α=0.02, effect size=-0.47) and not for experts. Meanwhile, the values "1" that appeared in Table 5 for efficiency and satisfaction of R3 in the Ar replication, show up in Table 9 for experts only, which appears to be the audience who benefits the most from this refactoring (M2: α=0.03, effect size=0.42; M3: α=0.007, effect size=0.46).

Table 9. Analysis by user experience

	Sp						**Ar**					
	Experts			**Novice**			**Experts**			**Novice**		
Ref. ID	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
R1	0	0	0	0	0	0	0	1	1	0	1	1
R2	0	0	0	0	0	0	0	0	0	0	0	0
R3	0	0	0	-1	0	0	0	1	1	0	0	0
R4	0	0	0	0	0	0	0	0	0	0	0	0
R5	0	0	0	0	0	0	0	0	0	0	1	0
R6	0	0	0	0	0	0	0	0	0	0	0	0
R7	0	0	0	0	-1	0	0	-1	0	0	0	0
R8	0	0	0	0	0	0	0	1	0	0	0	0
R9	0	0	0	0	0	0	0	1	0	0	0	-1
R10	0	0	1	0	1	0	0	1	1	0	1	0
R11	0	0	0	0	0	0	0	0	0	0	0	0
R12	0	0	0	0	0	0	0	0	0	0	0	0
R13	0	0	0	0	0	0	0	0	0	0	0	0
R14	0	0	0	0	0	0	0	0	1	0	0	0
R15	0	0	0	0	0	0	0	0	1	0	0	1
R16	0	0	0	0	0	0	0	0	0	0	0	0
R17	0	0	0	1	0	0	0	0	0	0	0	0
R18	0	0	1	0	0	0	1	0	1	0	1	0
R19	0	0	0	0	0	0	0	-1	0	0	0	0
R20	0	0	0	0	0	0	0	0	0	0	0	0
R21	0	0	0	0	-1	0	0	0	0	0	0	0

Nevertheless, there are other cases with values that differ from the preliminary analysis. The reason for this difference is probably due to a loss in statistical power. Since for this test we divide subjects into two groups depending on their expertise, we divide the number of experimental units by two. This reduction lowers the statistical power of the data (the probability of rejecting a false null hypothesis), which is a common phenomenon as reported by Dybå et al. (Dybå et al. 2006). One of these cases is R4 (*Change widget*), which lost its negative significance in efficiency in Ar, both for novices and experts. The other example is R5 (*Add a verification activity*), applied to insert a *captcha* in the registration form. This refactoring had no significance before and gained a positive significance in efficiency for novices in Ar (*p-value*=0.02, effect size=0.44). It may seem contradictory that inserting a *captcha* improves efficiency, but the reason again could be the reduction of subjects.

Concluding, this analysis showed that there are refactorings more suitable for novices (R5, R17, R19) and others more suitable for experts (R3, R8, R9, R10, R14, R18, R21), which amounts to almost half of the refactorings.

6.2.2 Type of e-commerce application

Since we use two problems (on-line stores and online auctions) and each problem is used in a different context, we also decided to analyze whether the type of context affects refactorings' significance. We conducted an analysis to discriminate the type of application, i.e., *Zencart* as an online store and *Webid* as an online auction, to highlight the differences between the kinds of activities in which each application differs, and how refactorings may have influenced one context in particular. Since the same subject did not interact twice with the same system, we have not used paired samples in this case but instead we applied Mann-Whitney U test. The effect size is also calculated with another formula: Z/√N, where Z is provided from the Mann-Whitney test and N is the sample size.

Table 10 shows the results of this analysis. The meaning of the values is the same as in Table 9. There are some refactorings that are only significant in one of both web applications. For instance, R3 in the Sp replication got a negative value in effectiveness only for *Zencart* (α=0.03, effect size=-0.45). There are other refactorings that present a different significance running the analysis classified by system, like the case of R2 (*Split page*) which shows four new positive significances: two of them for *WeBid* in efficiency (in Sp: α=0.01, effect size=-0.52; in Ar: α=0.009, effect size=-0.51), one for *WeBid* in satisfaction in Ar (α=0.04, effect size=-0.39), and one for *Zencart* in satisfaction in Ar (α=0.01, effect size=-0.5).

Table 10. Analysis by system

	Sp						**Ar**					
	Zencart			**Webid**			**Zencart**			**Webid**		
Ref. ID	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
R1	0	0	0	0	0	0	0	1	1	0	1	0
R2	0	0	0	0	1	0	0	0	1	0	1	1
R3	-1	0	0	0	0	0	0	1	1	0	0	0
R4	0	0	0	0	0	0	0	0	1	?	?	0
R5	?	?	0	0	-1	0	?	?	0	0	0	0
R6	0	0	0	0	0	0	0	0	0	0	0	0
R7	0	-1	0	0	0	0	0	-1	1	0	-1	0
R8	0	1	0	0	0	0	0	0	0	0	1	0
R9	0	1	0	0	0	0	0	0	0	0	1	0
R10	0	0	1	0	1	0	0	0	1	0	1	1
R11	0	0	0	0	0	0	0	0	1	0	-1	0
R12	0	0	0	0	0	0	0	-1	0	0	0	0
R13	0	0	0	0	0	0	0	-1	0	0	-1	0
R14	0	0	1	0	0	0	0	-1	1	0	0	0
R15	0	0	0	0	0	0	0	0	1	0	1	1
R16	0	0	0	0	0	0	0	0	0	0	0	0
R17	0	0	0	0	0	0	0	0	1	0	1	0
R18	0	0	0	0	0	1	1	1	1	0	1	0
R19	0	0	0	0	0	0	0	-1	0	0	0	0
R20	0	-1	0	0	0	0	0	0	0	0	0	0
R21	0	-1	0	0	0	0	0	0	0	0	0	0

Besides R2, there are four refactorings that got a new positive significance for *Zencart* in satisfaction in the Ar replication. These refactorings are: R4 (*Change widget*; α=0.002, effect size=-0.59), R7 (*Aggregate activities*; α=0.04, effect size=-0.4), R11 (*Move widget*; α=0, effect size=-0.71), and R17 (*Make explicit the steps composing a process and the current step being executed*; α=0.02, effect=-0.45). These differences show that some refactorings could be more interesting depending on the context of use or the implementation.

There are other refactorings that are significantly better in efficiency for the ¬R versions of the systems, for *Webid* (R5 in Sp and R11 in Ar), for *Zencart* (R12 and R14 in Ar, R20 in Sp), and for both applications in the case of R13 (only in the Ar replication). Thus, in this last case it does not appear to be a correlation with the type of application.

The entries in Table 10 that contain a question mark ("?") mean that we could not apply the statistical test because there was not enough data in at least one of the groups resulted from the classification of subjects per type of e-commerce application they interacted. This did not happen in earlier analyses because in the original analysis we had the largest sample size (no division per subjects or problems) and the division of subjects per experience resulted in groups with enough data to apply the statistical test.

Summarizing, we can state that there are differences in significant refactorings between both problems. These differences may arise due to two characteristics that differentiate them: the combination of refactorings and the context of use. Both problems have the same refactorings but they are not combined within the same interfaces in both problems. For example, in *WeBid*, R11 was implemented together with R13 to move actions like “Add to watch list” from the top of the page and distributing them between every element in a category list. Meanwhile, in *Zencart*, R11 was implemented on its own to move action buttons from the bottom to the top of a page. Looking at the results we can conclude that implementing R11 together with R13 was not beneficial, probably because R13 swallowed the benefits of R11. Some combinations of refactorings might obtain better results of usability than others.

Note also that the implementation of each refactoring in both systems is not always the same, as it depends on the context of use where each refactoring is applied. Significant results that are specific to only one problem show that there are refactorings more suitable than others for a specific context of use.

Refactorings that are significant for *Zencart* are more suitable for online stores systems while refactorings that are significant for *WeBid* are recommended for auctions systems.

6.2.3 Alternative treatment of null values

In order to avoid the threat *Maturation*, which means that subjects get bored as time goes, we added a “skip” button to allow subjects to interrupt the current activity and continue with the next one. Evidently, letting users skip activities was not beneficial for the experiment results, but we preferred avoiding the threat *Maturation*, mainly since the experiment was long. Since we used paired samples in our statistical test, further insight is required to check how the missing data may have influenced our analysis. Before processing the data, we filtered out the subjects with more than 10 incomplete activities. With this filter we made sure that each subject had 10 empty values at most. In our base analysis, we have processed the data with these empty metrics. Therefore, the preliminary analysis had to discard several pairs for cases where the subjects skipped activities and whose related variables got non-applicable (N/A) values.

In order to study whether or not these empty metrics are affecting our results, we have conducted a post-hoc analysis completing empty metrics with the most pessimistic values. That is, the analysis was calculated filling in empty cells with 0 for effectiveness and efficiency and 1 for satisfaction. The idea is that if an activity is skipped, this means that the degree of success (effectiveness) is 0%, which involves also an efficiency of 0 (calculated as effectiveness/time). Moreover, when an activity is skipped, we can guess that the subject was not satisfied during its execution; therefore we can assign the value 1 to satisfaction (i.e., the least value for satisfaction) to any skipped activity. Table 11 shows the results of the analysis (M1=*effectiveness in use*, M2=*efficiency in use*, M3=*satisfaction in use*). Again, the meaning of the values 1, -1 and 0 is the same than in previous tables.

Table 11. Optimizing empty cells

	Sp			**Ar**		
Ref. ID	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
R1	0	0	0	1	1	1
R2	0	1	0	1	0	0
R3	-1	0	0	0	1	1
R4	-1	-1	0	-1	-1	0
R5	1	0	0	1	1	0
R6	0	0	0	0	0	1
R7	0	0	0	0	-1	0
R8	0	0	0	0	1	0
R9	0	0	0	0	1	0
R10	0	1	1	0	1	1
R11	0	0	0	0	0	0
R12	0.65	0	0	0	0	0
R13	0.65	0	0	0	0	0
R14	0.19	0	0	-1	0	1
R15	0.66	0	0	0	0	1
R16	0.77	0	0	0	0	0
R17	0.46	0	0	0	0	0
R18	0.75	0	0	1	1	1
R19	0.7	0	0	0	0	0
R20	0.96	0	0	0	0	0
R21	0.37	0	0	0	0	1

Table 11 shows better results with regard to the base experiment in five cases:

- R2) *Split Page:* used in both systems to separate the login process from the first-time registration, and to split and organize the information of a product in separate tabs like "Description", "Technical details", "Shipping & payment". This refactoring got significant values to reject the null hypothesis in favor of the R version for efficiency in Sp (α=0.05, effect size=0.34) and for effectiveness in Ar (α=0.02, effect size=0.17).
- R5) *Add a verification activity*: applied to insert a *captcha* in registration forms, got enough significance to reject the null hypothesis in favor of the R version for effectiveness in both replications (Sp: α=0.002, effect size=0.45; Ar: α=0.0, effect size=0.48), and for efficiency in Ar (α=0.0, effect size=0.66).

- R7) *Aggregate activities*: used in both applications to aggregate in the same page the possibility of updating different sections of the personal account information. The significance in efficiency for the ¬R version was lost in this analysis in the Sp replication and only appears in Ar (α=0.001, effect size= -0.56).
- R19) *Add a summary activity*: this refactoring was applied in both applications to provide details of a purchase or a new auction before confirmation. The negative significance in efficiency in the Ar replication was lost in this analysis.
- R21) *Add an "assistance" activity*: it was used to add auto-complete in the search field of a product. This refactoring gained significance in satisfaction in Ar (α= 0.05, effect size=0.26), and lost its negative significance in efficiency in Sp.

Therefore, we conclude that the existence of empty cells affects the significance of some refactorings. Affected refactorings are those with most metrics with empty values (23.8% of all the refactorings), which is a short percentage from the amount of refactorings. The percentage of empty metrics for R2, R5, R7, R19 and R21 are shown in Table 12. A possible reason that justifies the lack of values for some metrics of these refactorings is the difficulty of the activities involved in their analysis. Difficult activities were skipped by the most of the novice users.

Table 12. Percentage of metrics without value

	Sp Replication			Ar Replication		
	Effectiveness	**Efficiency**	**Satisfaction**	**Effectiveness**	**Efficiency**	**Satisfaction**
R2	2.3%	2.3%	2.3%	1.9%	1.9%	1.9%
R5	27.3%	27.3%	0%	27.8%	27.8%	1.9%
R7	6.8%	6.8%	6.8%	5.6%	5.6%	5.6%
R19	13.6%	27.3%	15.9%	11.1%	22.2%	11.1%
R21	6.8%	6.8%	6.8%	6.8%	6.8%	6.8%

6.3 Analysis of implementation effort

We recruited 3 students with different background and expertise to play the role of developers. They were in charge of applying the refactorings for both web applications used in the experiment. We asked the developers to rate (as soon as they'd finished implementing each refactoring) the effort spent in their implementation as *easy*, *medium*, or *hard*, and then we averaged the results. These measures provide an indication of the effort required to implement each refactoring, regarding time and difficulty. We show a complete list of refactorings and their effort in Table C.1 in Appendix C. The table shows that for most refactorings, the effort to implement them was medium, about a quarter of them were easy, and only 3 of them were considered hard. We observed that the hardest ones involved complex navigation restructuring that required a deeper knowledge on the application's implementation (or the used framework), as opposed to just adding navigation steps or replacing widgets. However, even if altering the steps in a navigation-based process can bring great benefits (as observed in the results of R17, R18 and R20), making sure that these steps are easier to follow will also improve these processes at a lower cost (as seen in results for R3, R6, R10, R14).

Even if the sample of developers is very small, having two applications gave us the chance of evaluating each refactoring in two different contexts. This reduced, to some degree, the bias of considering the specifics of each application, instead of the difficulty of implementing a given refactoring. Hence, we planned the development in such a way that each developer implemented different refactorings on both applications. In order to mitigate the learning effect, we also made sure they did not implement the same refactoring twice.

Furthermore, by using both implementation effort and level of usability improvement, we were able to assign a priority to refactorings, in order to help developers choose among them to provide the most value in the least amount of time. This is discussed in the next section.

6.4 Discussion

The previous subsections provide a broad range of results of our experiment under different kinds of analysis. Summarizing, the results show the particular situations in which refactorings are more useful in improving usability:

- There are refactorings that always improve usability (R10, R14, R18);
- There are refactorings that are more useful for those with knowledge of software development (R1, R3, R6, R8, R9, R15);

- Other refactorings are more useful for those without knowledge of software development (R17, R20);
- There are refactorings that are more useful for those who use e-commerce sites frequently (R3, R8, R9, R10, R14, R18, R21), while other refactorings are better for sporadic users of e-commerce sites (R5, R17, R19);
- Some refactorings are better in the context of online stores (R4, R7, R11), and other refactorings are better for online auctions (R2, R3).

The lack of significance independent of the type of user or the type of context in most refactorings can be the result of different aspects. First, the possibility to skip an activity when the subject does not know how to finish it reduces the threat of getting bored (*Maturation*). However, this results in a few metrics without values for some subjects. These empty metrics might be affecting results for R2, R5, R7, R19 and R21, since we have demonstrated that they are significant if we fill in empty metrics with the most pessimistic values. Second, some refactorings appear in more than one activity, which involves an aggregation of metrics per refactoring. Effectiveness, efficiency and satisfaction are aggregated through their average, which might involve a significance loss. Third, refactorings can only be checked with users through a specific implementation in a web application. The way in which each refactoring is implemented might be affecting the significance of the results. For example, refactoring *Split Page* depends deeply on the way contents are distributed; in the case of *Change widget* we used a calendar for a birthday input, and other options might have been more suitable, like displaying multiple fields or select boxes for *day*, *month* and *year*. Fourth, we have focused our experiment on a reduced number of subjects during their interaction with two web applications. An increase in the number of subjects and the replication of the experiment with other problems might increase the amount of refactorings that significantly improve usability in use. Fifth, combining several refactorings for the same activity. In the experiment we had a total of 8 activities that involved two or more refactorings. In most cases, the combination gave way to a single, larger change in the application, like the registration activity in both problems, which involved 4 refactorings in the *Zencart* site and 2 refactorings in the *WeBid*. Naturally, some of these refactorings received shared scores with the others involved in the same activities - for instance, in Zencart's registration activity, the aforementioned implementation of R4 (replace widget applied to date input) might have negatively affected the whole registration activity in the efficiency aspect, but also might have received a positive score influence from R3 (validation) in the same aspect. While we do believe a set of refactorings to improve an activity has a positive synergic effect, we also think that a more comprehensive experiment is required to confirm this, where all refactorings are evaluated both on their own, and together with other refactorings.

The design we have used in our experiment offers some strong points. First, the replication with subjects of different profiles allows studying the refactorings that are independent of end-users' profile. Moreover, we can identify the refactorings that are recommended for the context of online stores and which ones are recommended for online auctions. Second, in each replication, we have mainly two roles of subjects: novices and experts. We have analyzed which refactorings are more beneficial for novices and which ones are more beneficial for experts. This idea is aligned with statements of the Human Computer Interaction (HCI) community about providing personalized features depending on user's profiles (Van Welie & Trætteberg 2000). In the future, systems are tending to be as adaptive as possible to users' profiles, so the use of refactorings can be customizable depending on the type of user. Third, analyzing all the alternatives we used, only three refactorings remain without a positive significance in at least one dependent variable:

- R12) *Split list* and R13) *Distribute menu* showed that the ¬R version was significantly better than the R version in the Ar replication during the analysis by system. R12 was applied to both applications to divide the alphabetical list of products in a category of four pages, so subjects did not need to scroll excessively to find a product, they could jump directly to the needed page. R13 was applied to place an action that subjects had to execute, in the list of products besides each one.

  We noticed that both refactorings had participated in activities together with other refactorings: Zencart's activity 4.2 (see Appendix A) encompassed refactorings R12, R13 and R14, while WeBid's activity 4.6 involved refactorings R11 and R13. In this last activity, subjects had to find and perform an action over about 4 to 5 different products. The negative significance may have had two reasons. Firstly, we realized after the experiment that the activity was probably too long, and since it was among the last activities, subject may have been tired. Second, R12 made the activity harder as the products to select were too few to make split pages worthy. In that case, scrolling in the page was easier. Also, having more than one refactoring per activity makes it difficult to know which refactoring of all the ones included in the activity is affecting the outcomes.

- R16) *Remove duplicated process link*, involved in WeBid’s activities 2.2 and 2.5, and Zencart’s activity 4.6 did not show any significance. This outcome is not really surprising because link duplication was not annoying or confusing in the particular applications that we used.

From all the analyses presented in this article, we were able to deduce a prioritized list of refactorings that developers may use to systematically improve usability on their sites. The strategy we selected to order refactorings is as follows:
1st) By number of significances (i.e., quantity of "1" values in a row of Table 5);
2nd) By number of different dependent variables that got significances;
3rd) By level of implementation effort, from easy to hard;
4th) By effect size.
From this ordering strategy we obtain the prioritized list depicted in Table 13. This classification can enhance the development effort, discarding non-significant refactorings or those difficult to implement.

Table 13. List of refactorings sorted by priority

Ref. ID	Name
R10	Provide breadcrumbs
R18	Enable user to go back in the process steps
R1	Improve the description of process links
R17	Make explicit the steps composing the process and the current step being executed
R14	Keep the user informed on the ongoing process
R3	Anticipate a validation activity
R6	Add a "confirm and commit" activity
R15	Improve link destination announcement
R20	Add processing page
R8	Introduce information on demand
R9	Turn attribute into link
R2	Split page
R5	Add a verification activity
R11	Move widget
R4	Change widget
R21	Add an "assistance" activity
R16	Remove duplicated process links
R12	Split list
R19	Add a summary activity
R13	Distribute menu
R7	Aggregate activities

7. CONCLUSIONS

This article presented the first experiment conducted to measure the real benefits of applying a set of refactorings to improve usability of e-commerce applications.

When developers apply internal refactorings that make their job easier, they have better chances of selecting the best refactorings to solve their needs. When the refactorings improve an external quality attribute, which depends on end-users, selecting the right refactorings and prioritizing the transformation process is much more complicated. In this sense, this work analyses the results from different perspectives and contexts, to help developers in the right selection of refactorings.

Results show that the concept of usability is too complex and the benefits of each refactoring cannot be simplified to a single classification between right and wrong. The degree of improvement on usability carried by each refactoring, as a subjective characteristic, depends on the type of users and the type of system. Nevertheless, it is important to note that there were no refactorings with significant negative results in satisfaction. This allows us to conclude that usability is enhanced or preserved by the refactorings evaluated in this work, which is a very strong point in favor of usability refactorings.

Besides the valuable results presented in this article, there are side contributions of this work related to the experiment design and procedure. Since this was the first experiment of this kind, we had to plan it from scratch and decide the most appropriate usability metrics, define tasks and activities to exercise each refactoring, create questionnaire to measure satisfaction, and build a tool that automates most of the experiment procedure, helping subjects in executing tasks and us in collecting measurements. All of these can be reused in another experiment of the same characteristics. The resources for conducting the experiment are available online[1].

REFERENCES

Barnes, S. & Vidgen, R., 2000. WebQual: An Exploration of Web-Site Quality. In *Proceedings of the eighth European conference on information systems*. pp. 298–305.

Barnes, S.J. & Vidgen, R., 2003. Measuring Web site quality improvements: a case study of the forum on strategic management knowledge exchange. *Industrial Management & Data Systems*, 103(5), pp.297–309.

Basili, V.R., Caldiera, G. & Rombach, H.D., 1994. The goal question metric approach. *Encyclopedia of Software Engineering*, 1, pp.528–532.

Benjamini, Y. & Hochberg, Y., 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57, pp.289 – 300.

Blaikie, N., 2003. *Analyzing quantitative data: From description to explanation*, Sage Publications.

Bruun, A. et al., 2014. Active Collaborative Learning: Supporting Software Developers in Creating Redesign Proposals. In Springer Berlin Heidelberg, ed. *Human-Centered Software Engineering*. pp. 1 – 18.

Cohen, L., Manion, L. & Morrison, K., 2007. *Research Methods in Education*,

Dearden, A. & Finlay, J., 2006. Pattern languages in HCI: A critical review. *Human–computer interaction*, 21(1), pp.49–102.

Dig, D., 2011. A refactoring approach to parallelism. *IEEE Software*, 28(1), pp.17–22.

Distante, D. et al., 2014. Business Processes Refactoring to Improve Usability in E-Commerce Applications. *Electronic Commerce Research*, 14(4), pp.1–42.

Van Duyne, D., Landay, J. & Hong, J., 2007. *The design of sites: Patterns for creating winning web sites*, Prentice Hall Professional.

Dybå, T., Kampenes, V.B. & Sjøberg, D.I.K.K., 2006. A systematic review of statistical power in software engineering experiments. *Information and Software Technology*, 48(8), pp.745–755.

[1] http://selfrefactoring.s3.amazonaws.com/guide_tool/instructions.html

Faul, F. et al., 2007. G* Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior research methods*, 39(2), pp.175–191.

Fernandez, A., Insfran, E. & Abrahão, S., 2011. Usability evaluation methods for the web: A systematic mapping study. *Information and Software Technology*, 53(8), pp.789–817.

Fernández-Ropero, M. et al., 2012. Quality-Driven Business Process Refactoring. In *International Conference on Business Information Systems (ICBIS 2012)*. pp. 960–966.

Fowler, M. & Beck, K., 1999. *Refactoring: improving the design of existing code*, Addison-Wesley.

Garrido, A. et al., 2013. Personalized web accessibility using client-side refactoring. *IEEE Internet Computing*, 17(4), pp.58–66.

Garrido, A., Rossi, G. & Distante, D., 2011. Refactoring for Usability in Web Applications. *IEEE Software*, 28(3), pp.60–67.

Garrido, A., Rossi, G. & Distante, D., 2009. Systematic Improvement of Web Applications Design. *Journal of Web Engineering*, 8(4), pp.371–404.

Grissom, R.J. & Kim, J.J., 2005. *Effect Sizes For Research: A Broad Practical Approach*, Taylor & Francis Group.

Harold, E., 2008. *Refactoring html: improving the design of existing web applications*, Addison-Wesley Professional.

ISO, I., 2011. ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models.

Jönsson, P. & Wohlin, C., 2004. An evaluation of k-nearest neighbour imputation using lIkert data. In *Proceedings - International Software Metrics Symposium*. pp. 108–118.

Juristo, N. & Moreno, A., 2010. *Basics of software engineering experimentation*, Springer Publishing Company, Incorporated.

Kim, M., Zimmermann, T. & Nagappan, N., 2012. A Field Study of Refactoring Challenges and Benefits. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering - FSE '12*. ACM, p. 50.

Maruyama, K., 2007. Secure Refactoring - Improving the Security Level of Existing Code. In *ICSOFT 2007, Proceedings of the Second International Conference on Software and Data Technologies, Volume SE, Barcelona, Spain, July 22-25, 2007*. pp. 222–229.

Moody, D. et al., 2003. Evaluating the quality of information models: empirical testing of a conceptual model quality framework. *Proceedings of the 25th International Conference on Software Engineering*, pp.295–305.

Murphy-Hill, E., Parnin, C. & Black, A.P., 2012. How we refactor, and how we know it. *IEEE Transactions on Software Engineering*, 38, pp.5–18.

Nanard, M., Nanard, J. & Kahn, P., 1998. Pushing Reuse in Hypermedia Design: Golden Rules, Design Patterns and Constructive Templates. In *Proceedings of the ninth ACM conference* *on Hypertext and hypermedia : links, objects, time and space---structure in hypermedia systems links, objects, time and space---structure in hypermedia systems - HYPERTEXT '98*. New York, New York, USA: ACM Press, pp. 11–20.

Nielsen, J., 1999. *Designing Web Usability*, New Riders Publishing.

Nielsen, J. & Tahir, M., 2002. *Homepage usability: 50 websites deconstructed*, New Riders.

Olsina, L. et al., 2008. Web Applications Refactoring and Evaluation: A Quality-Oriented Improvement Approach. *Journal of Web Engineering*, 7(4), pp.258–280.

Rieger, M. et al., 2007. Refactoring for performance: an experience report. *Proc. Software Evolution*, 2, p.9.

Shneiderman, S. & Plaisant, C., 2005. Designing the user interface 4 th edition. *ed: Pearson Addison Wesley, USA*.

Vakilian, M. et al., 2012. Use, disuse, and misuse of automated refactorings. In *Proceedings - International Conference on Software Engineering*. pp. 233–243.

Van Welie, M. & Trætteberg, H., 2000. Interaction patterns in user interfaces. *7th Pattern Languages of Programs Conference*, 13, p.16.

Wnuk, K., Gorschek, T. & Zahda, S., 2013. Obsolete software requirements. *Information and Software Technology*, 55, pp.921–940.

Wohlin, C. et al., 2012. *Experimentation in software engineering: an introduction*, Springer.

Ying, M. & Miller, J., 2013. Refactoring legacy AJAX applications to improve the efficiency of the data exchange component. *Journal of Systems and Software*, 86(1), pp.72–88.

Zibran, M.F. & Roy, C.K., 2011. A constraint programming approach to conflict-aware optimal scheduling of prioritized code clone refactoring. In *Proceedings - 11th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2011*. pp. 105–114.

Zou, Y., Zhang, Q. & Zhao, X., 2007. Improving the usability of e-commerce applications using business processes. *IEEE Transactions on Software Engineering*, 33(12), pp.837–855.

APPENDIX A

This appendix lists, for each of the websites used in the empirical study, the user tasks, component activities, and refactorings applied to each activity of the experiment.

Zencart-based online-store

User Task	Component Activities	Refactorings tested with the activity
Task 1. Register to the website	1. Access the user registration page	R1 - *Improve the description of process links* (having the "Login" link to navigate only to the Login page, and not to the registration)
		R2 - *Split page* (separating the "Login" page from the "Register" one)
	2. Fill in the user registration form with provided data (in particular, shipping and billing address must be the same)	R3 - *Anticipate a validation activity* (validating date of birth, equality of both password fields, and email format, as soon as the focus moves out of the correspondent field)
		R4 - *Change widget* (using a javascript calendar to fill in the date of birth)
		R5 - *Add a verification activity* (adding a captcha at the end of the registration form)
		R6 - *Add a "confirm and commit" activity* (summarizing all entered data and presenting it in a page asking for confirmation or otherwise return to the form)
	3. Finalize the registration (click on submit)	
	4. Log Out	
Task 2: Update user registration info	1. Log in into the website	
	2. Update your account info: • Change billing address • Change your password • Add a second shipping address	R7 - *Aggregate activities* (having in one page the activities to: view or change account information, view or change entries in address book, change account password)
		R3 - *Anticipate a validation activity* (anticipating the validation of: date of birth, "confirm password" checking that it is equal to the field "password", and email format
	3. Finalize the process (click submit)	
	4. Log out	
Task 3: Search for a number of products in the catalog of the store	1. Log in into the website	
	2. Search for "Mouse" and check if any "Microsoft" mouse is available and visit the page of the product	R21 - *Add an "assistance" activity* (adding the autocomplete feature to the search textbox)

	3. Check the technical information on the "Matrox G400" product and make sure it is compatible with Ubuntu (DO NOT USE SEARCH TOOL)	R2 - *Split Page* (splitting product information into different sections that can be navigated by tabs: "Description", "Technical info", "Shipping Costs".)
	4. Check the average ranking of reviews available for the "Matrox G400" product and read the review which ranks the product "2 stars".	R8 - *Introduce Information on demand* (instead of having the review's graphic at the end of the page, showing the number of reviews for each ranking value when hovering over "Reviews" at the top)
		R9 - *Turn attribute into link* (turning each ranking value into a link that navigates to the page showing the reviews on the product, and specifically, to the point where reviews on that ranking value start)
	5. Check which are the other products available in the same category of the "Matrox G400" (DO NOT USE THE BACK button of the browser)	R10 - *Provide breadcrumbs* (for category)
	6. Log out	
Task 4: Make a purchase with several products	1. Enter the category "Hardware --> Graphics Cards", view info on the product "Matrox G200 MMS", choose the version of the product with 16 MB of memory and add 2 items of it to your shopping cart	R11 - *Move widget* (changing the position of "Quantity" textbox, "Add to cart" button and the "Please Choose" group to make them visible without scrolling the page)
	2. Enter the category "DVD Movies --> Action", search in the list of products the titles "The Matrix Linked" and "Under Siege Linked", "Fire Down Below Linked", "Speed Linked", and add them to your shopping cart, directly from the list, without viewing the details on each DVD.	R12 - *Split List* (diving the list of movies, organized in alphabetic order, by ranges of letters "A-D", "E-I", "L-O", "P-Z", with links to each group at the top of the page)
		R13 - *Distribute menu* (Distribute the "Add selected products to cart" button to each of the product thumbnails shown in a category changing the text into "Add to shopping cart")
		R14 - *Keep the user up to date on the ongoing process* (showing an overlay message saying that the product has actually been added to the cart)
	3. Enter the category "Hardware --> Printers", view detailed info for printer "EPSON MX14" and view the image of the printer in detail	R4 - *Change widget* (change the widget used to show the "Larger image "of the product with a RIA one enabling to zoom the image and see details of it in the same page)
	4. Add two items of the EPSON MX14 printer to the shopping cart	R11 - *Move widget* (change the position of "Quantity" textbox, "Add to cart" button to make them visible without scrolling the page)
	5. Check the number of items and total cost of products in the shopping cart	R15 - *Improve link destination announcement* (about the shopping cart)
	6. Visit the shopping cart and remove "The Matrix Linked" DVD and add one	R16 - *Remove duplicated process links* (to remove products of the shopping cart)

	more item of the "Under Siege Linked" DVD	
	7. Start the checkout process	R17 - *Make explicit the steps composing a process and the current step being executed* (adding the steps of the checkout process and highlighting the current step)
	8. Select as shipping address the second address in your address book and as shipping method the cheapest one, then continue checkout	R7 - *Aggregate activities* (inserting the "Change address" activity into the "Shipping information " one, by enabling the user to select one of the addresses in its address book, without the need to navigate to the address book webpage)
	9. Add payment info by entering credit card data	R3 - *Anticipate a validation activity* (for credit card data form)
	10. Go back in the process to change your shipping address to be the first of your address book	R18 - *Enable the user to go back in the process steps* (turning the process steps shown on top of the page to be clickable and allow navigation to each one)
	11. Go ahead in the process to the Order Confirmation page and verify that you are purchasing the correct products and quantities	R19 - *Add a summary activity* (with information on the shopping cart before confirm)
	12. Confirm the order by pushing the "Confirm the order" button	R20 - *Add processing page* (having a "Processing page" to inform the user that the order is being stored and inviting him to wait)
	13. Click the OK button in the confirmation page you receive after finalizing your order	

WeBid-based online auction

User Task	Component Activities	Refactorings tested with the activity
1. Register to the website	1. Access the user registration page	R1 - *Improve the description of process links* (having the "Login" link to navigate only to the Login page, and not to the registration)
		R2 - *Split page* (separating the "Login" page from the "Register" one)
	2. Fill in the user registration form by specifying provided data	R3 - *Anticipate a validation activity* (validating date of birth, username already exists, equality of both password fields, and email format, as soon as the focus moves out of the correspondent field)
		R5 - *Add a verification activity* (adding a captcha at the end of the registration form)
	3. Verify the "Terms and Conditions" and submit the registration form	R9 - *Turn attribute into link* (from the phrase “I agree to the terms and conditions” to the actual terms and condition statement.
	4. Activate your user account (check email and press “Activate me”) to finalize the registration	

2. Update your user registration info	1. Log in into the website	
	2. Update your account info: a. change your address b. Change your password c. Subscribe to newsletter d. Change payment method	R16 - *Remove duplicated process links* (so the “My control panel” menu is not duplicated)
		R7 - *Aggregate activities* (having in one page the activities to: change account password, change address and basic info, subscribe to newsletter and payment info)
	3. Update your date of birth	R4 - *Change widget* (using a Javascript calendar to enable the user modify his date of birth)
	4. Confirm changes	R6 - *Add a “confirm and commit” activity*
	5. Finalize the process (click ”Save changes”)	R16 - *Remove duplicated process links* (for the logout link)
	6. Log out	
3. Search for a number of items in the auction website	1. Log in into the website	
	2. Search for "guitar" and check if a "Fender" guitar is available	R21 - *Add an "assistance" activity* (adding the autocomplete feature to the search textbox)
	3. Check the payment and shipping information on the "Fender guitar" and check (i) if the seller ships internationally (ii) if the seller accepts Cash on delivery (DO NOT USE THE SEARCH TOOL)	R2 - *Split Page* (diving information into different sections that can be navigated by tabs: "Item description", "Picture gallery", “Shipping & Payment”)
	4. Check how many active auctions the seller of the “Fender guitar” has	R8 - *Introduce information on demand* (when the user hovers the mouse over the “View active auctions” link, pop-up some of the information without needing to navigate)
	5. Navigate to the home page of the website, look for an auction on a Nikon D4 camera and access the web page showing the details on the bid	R9 - *Turn attribute into link* (so that the pictures of the items are clickable)
	6. Check if the camera has a shoe mount for an external flash	R4 - *Change widget* (change the widget used to show the "Larger image "of the product with a RIA one enabling to zoom the image and see details of it in the same page)
	7. Check which are the other products available in the same category of the "Nikon D4" (DO NOT USE THE BACK button of the browser)	R10 - *Provide breadcrumbs*
	8. Log out	

4. Make a bid on a iPad	1. Enter the category “Electronics & Photography”, search for an auction on the iPad with 16 GB of memory and visit the product details web page	R12 - *Split List* (diving the list of items, organized in alphabetic order, by ranges of letters "A-D", "E-I", "L-O", "P-Z", with links to each group at the top of the page)
	2. Find out the amount of the current bid in dollars.	R15 - *Improve link destination announcement* (where it says the unit of the price, e.g. EUR, show an image of a calculator to better announce that the link goes to a currency converter)
	3. Place a bid for 90 EUR first, and then another for 110 EUR on the iPad 16GB	R3 - Anticipate a validation activity (anticipating the validation of the amount to be higher than the current highest bid)
	4. Find out the status of your bid	R14 - *Keep the user informed on the ongoing process* (showing the “You’re the highest bidder sign”.)
	5. Find out what an "Item watch" means by checking it in the page: Help --> Buying	R16 - *Remove duplicated process links* (for the Help menu)
	6. Add to your watch list every item in the category "Home & Garden"	R11 - *Move widget* R13 - *Distribute Menu* (for the actions “Add to watch list” and “Send to a friend”)
	7. Log out	
5. Sell a DVD	1. Suppose you want to sell a DVD into the auction website. Log into your account and proceed to sell an item	R17 - *Make explicit the steps composing a process and the current step being executed* (adding the steps of the process -- “Add product details”, “Specify payment and shipping info”, "Confirm auction" -- and highlighting the current step)
	2. Select the category “Music & Video" --> "DVD" for the item you want to sell and go ahead to the next step in the process	R15 - *Improve link destination announcement* (replacing the text of the button from “Select category” to “Proceed to next step”.)
	3. Specify “Speed linked” as title of the DVD you want to sell, “DVD Regional Code: 2” as item description and select "Wire Transfer" as payment method. Then complete the process	R2 - *Split Page* (to enter all data on the auction, Payment and shipping will be on a different page)
	4. Go back in the process and change the category of your item to “Music & Video" --> “HD-DVD”	R18 - *Enable the user to go back in the process steps* (turning the process steps shown on top of the page to be clickable and allow navigation to each one)
	5. Change the item description by adding a picture for it (use the picture provided)	R20 - *Add processing page* (having a "Processing page" to inform the user that the picture is being uploaded and inviting him to wait)
	6. Select “Paypal” as payment method, shipping fee should be 20 and specify “Seller pays shipping expenses”	(Related to previous refactorings “Make explicit the steps composing a process” and “Split page”

	then submit the form	
	7. Confirm your auction	R19 - Add a summary activity (showing the current version of the auction confirmation page, which shows a summary of the auction data)
		R1 - *Improve the description of process links* (the page showing the summary of the auction data and asking for a confirmation will show two buttons: “Confirm auction” and “Cancel” instead of “Submit auction”.)
	8. Log out	

APPENDIX B

Table B.1 Summary of descriptive data

		AR					SP				
Refactoring	**Response Variable**	**Number of valid data**	**Minimum**	**Maximum**	**Mean**	**Median**	**Number of valid data**	**Minimum**	**Maximum**	**Mean**	**Median**
R1	Effectiveness ¬R	22	100	100	84,62	100	18	100	100	81,82	100
	Effectiveness R	27	100	100	100	100	21	100	100	95,45	100
	Efficiency ¬R	23	0,31	10,36	3,36	2,94	18	0,42	12,12	3,06	2,1
	Efficiency R	27	1,34	13,48	6,83	7,33	21	1,11	25,77	5,84	3,42
	Satisfaction ¬R	26	1	5	3,12	3	22	1	5	3,48	3,5
	Satisfaction R	27	1,5	5	4,04	4,5	22	2	5	3,78	4
R2	Effectiveness ¬R	25	33,33	100	91,33	100	21	33,33	100	91,27	100
	Effectiveness R	27	80	100	97,78	100	21	50	100	90,71	100
	Efficiency ¬R	27	0,3	4,02	1,58	1,31	22	0,19	4,52	1,06	0,84
	Efficiency R	27	0,65	3,44	1,73	1,79	21	0,47	6,82	1,54	1,19
	Satisfaction ¬R	26	2,33	5	3,83	3,75	21	1,83	5	4,09	4
	Satisfaction R	26	1,62	5	4,31	4,48	22	2,56	5	4,23	4,06
R3	Effectiveness ¬R	26	87,5	100	97,6	100	22	50	100	94,89	100
	Effectiveness R	27	50	100	97,22	100	22	42,85	100	83,2	100
	Efficiency ¬R	27	0,49	1,72	1,07	1	22	0,47	2,96	0,99	0,86
	Efficiency R	27	0,37	2,9	1,45	1,37	22	0,28	3,04	1,07	0,89
	Satisfaction ¬R	26	1	5	3,18	3	22	2	5	4,13	4
	Satisfaction R	27	2	5	4,08	4,16	22	2,16	5	4,09	4,16
R4	Effectiveness ¬R	25	50	100	98	100	18	100	100	90	100
	Effectiveness R	12	100	100	100	100	10	100	100	100	100
	Efficiency ¬R	27	0,22	7,51	3,28	3,19	20	0,78	11,76	3,61	2,72
	Efficiency R	27	1,25	5,47	2,36	2,13	22	1,55	7,24	3,26	2,85
	Satisfaction ¬R	26	1	5	3,39	3,58	22	2,66	5	3,78	3,83
	Satisfaction R	27	2	5	3,81	4	22	2	5	3,85	4
R5	Effectiveness ¬R	12	100	100	100	100	10	100	100	100	100
	Effectiveness R	25	100	100	92,59	100	20	100	100	90,91	100
	Efficiency ¬R	27	0,28	31,95	5,37	2,68	22	0,64	40,98	14,65	12,34

	Efficiency R	25	0,86	17,99	6,36	5,4	20	0,91	32,79	7,13	5,66
	Satisfaction ¬R	26	1	5	3,27	3	22	1	5	3,5	3,5
	Satisfaction R	27	1	5	3,52	3	22	2	5	3,59	4
R6	Effectiveness ¬R	15	100	100	62,5	100	16	100	100	76,19	100
	Effectiveness R	22	100	100	84,62	100	17	100	100	89,47	100
	Efficiency ¬R	18	0,92	28,74	5,22	0	17	0,5	18,32	5,84	3,55
	Efficiency R	23	0,4	17,36	3,78	2,87	20	0,26	17,48	2,94	1,7
	Satisfaction ¬R	24	2	5	3,83	3,75	21	1	5	3,4	4
	Satisfaction R	26	2,5	5	4,44	4,5	19	2	5	3,68	4
R7	Effectiveness ¬R	25	75	100	99	100	20	50	100	89,58	100
	Effectiveness R	26	83,33	100	98,72	100	21	20	100	81,9	100
	Efficiency ¬R	27	0,71	4,12	2,08	1,84	22	0,08	5,7	1,6	1,27
	Efficiency R	27	0,44	1,79	1,08	1,07	22	0,14	7,25	1,27	0,88
	Satisfaction ¬R	25	1,25	5	3,85	4	20	2	5	4,17	4,25
	Satisfaction R	26	2,5	5	4,14	4,5	21	1,16	5	4,01	4,5
R8	Effectiveness ¬R	25	100	100	100	100	19	100	100	95	100
	Effectiveness R	26	100	100	100	100	21	100	100	100	100
	Efficiency ¬R	27	0,31	6,29	2,21	2,22	21	0,16	9,78	1,95	1,7
	Efficiency R	27	0,75	12,45	3,28	2,42	22	0,56	12,94	3,07	2,09
	Satisfaction ¬R	25	1	5	3,42	4	20	1	5	3	3
	Satisfaction R	26	1	5	3,67	3,5	21	2	5	3,76	4
R9	Effectiveness ¬R	26	100	100	100	100	20	100	100	95,24	100
	Effectiveness R	26	50	100	96,15	100	21	50	100	90,91	100
	Efficiency ¬R	27	0,28	6,75	2,32	2,15	21	0,16	15,48	3,26	1,82
	Efficiency R	27	0,75	10,91	3,3	2,58	21	0,56	12,94	3,59	2,99
	Satisfaction ¬R	26	1,5	5	4,02	4	21	1	5	3,88	4
	Satisfaction R	26	1	5	3,99	4	22	2	5	4,11	4
R10	Effectiveness ¬R	23	100	100	92	100	18	100	100	90	100
	Effectiveness R	26	100	100	100	100	19	100	100	90,48	100
	Efficiency ¬R	25	0,05	6,25	1,99	1,66	20	0,32	12,87	2,26	1,56
	Efficiency R	27	0,99	12,35	4,48	3,68	20	1,41	17,48	5,1	4,34
	Satisfaction ¬R	25	1,5	5	2,58	2,5	20	1	5	2,7	2
	Satisfaction R	26	1,5	5	3,56	3,75	21	2	5	3,45	3
R11	Effectiveness ¬R	24	100	100	96	100	19	100	100	100	100
	Effectiveness R	21	100	100	95,45	100	18	100	100	100	100

	Efficiency ¬R	26	0,43	17,7	2,3	1,5	22	0,24	4,41	1,62	1,6
	Efficiency R	26	0,45	4,66	1,98	1,8	22	0,26	3,69	1,3	1,15
	Satisfaction ¬R	25	1	5	3	3	19	1	5	3,68	4
	Satisfaction R	22	1	5	3,82	4	18	1	5	3,83	4,5
R12	Effectiveness ¬R	22	100	100	88	100	18	100	100	94,74	100
	Effectiveness R	19	100	100	86,36	100	17	100	100	94,44	100
	Efficiency ¬R	24	0,64	7,79	2,45	1,53	21	0,5	5,22	2,04	1,3
	Efficiency R	24	0,54	19,31	3,45	1,1	21	0,69	8,67	2,92	1,77
	Satisfaction ¬R	25	1	5	3,88	4	19	1	5	4,37	5
	Satisfaction R	22	1	5	4,14	4	18	2	5	4,44	5
R13	Effectiveness ¬R	24	100	100	96	100	19	100	100	100	100
	Effectiveness R	22	100	100	100	100	18	100	100	100	100
	Efficiency ¬R	26	0,43	17,7	1,87	1,3	22	0,24	4,41	1,19	1,12
	Efficiency R	27	0,45	4,66	1,49	1,19	22	0,26	1,8	1,04	1,09
	Satisfaction ¬R	25	1	5	3,28	3	19	1	5	3,58	4
	Satisfaction R	22	1	5	3,32	3	18	1	5	3,94	5
R14	Effectiveness ¬R	25	100	100	100	100	19	50	100	97,37	100
	Effectiveness R	21	100	100	95,45	100	18	50	100	91,67	100
	Efficiency ¬R	27	0,58	2,58	1,33	1,37	22	0,44	3,2	1,23	1,13
	Efficiency R	26	0,54	4,13	1,42	1,06	22	0,13	2,86	1,03	0,89
	Satisfaction ¬R	25	1	5	2,86	3	19	1	5	2,84	3
	Satisfaction R	22	2	5	4,11	4,5	18	1	5	3,75	4
R15	Effectiveness ¬R	25	50	100	98	100	19	50	100	92,11	100
	Effectiveness R	24	100	100	100	100	17	100	100	94,44	100
	Efficiency ¬R	27	0,61	35,46	4,14	2,66	22	0,32	9,49	2,89	1,79
	Efficiency R	27	0,79	15,38	4,7	3,96	21	0,48	10,75	3,26	2,81
	Satisfaction ¬R	25	0,66	5	2,45	2,5	19	0,5	4	2,42	2,5
	Satisfaction R	24	2	5	3,42	3	18	1	5	3,28	3
R16	Effectiveness ¬R	25	100	100	100	100	19	66,66	100	94,74	100
	Effectiveness R	24	100	100	100	100	18	50	100	92,11	100
	Efficiency ¬R	27	1,11	5,9	2,91	2,92	22	0,26	21,6	3,43	2,49
	Efficiency R	27	0,75	4,47	2,43	2,27	21	0,48	36,9	3,96	2,31
	Satisfaction ¬R	25	4	5	4,44	4	18	1	5	3,78	4
	Satisfaction R	24	2	5	4,46	5	18	3	5	4,56	5
R17	Effectiveness ¬R	24	33,33	100	90,97	100	18	50	100	84,21	100

	Effectiveness R	24	100	100	100	100	18	100	100	100	100
	Efficiency ¬R	27	0,5	19,65	8,64	7,27	21	0,37	22,52	7,17	4,39
	Efficiency R	27	1,62	22,62	10,27	9,54	22	1,05	20,66	11,17	11,71
	Satisfaction ¬R	24	1	5	3,81	4	19	2	5	3,38	3,5
	Satisfaction R	24	2	5	4,06	4	18	2	5	3,94	4
R18	Effectiveness ¬R	21	33,33	100	68,05	66,66	17	33,33	100	76,31	100
	Effectiveness R	24	66,66	100	98,61	100	18	33,33	100	85,18	100
	Efficiency ¬R	24	0,53	5,35	2,21	2,16	20	0,34	7,48	2,31	2,05
	Efficiency R	27	0,84	12,61	4,66	3,79	22	0,54	9,78	3,83	2,83
	Satisfaction ¬R	24	0,5	5	2,63	2,5	19	0,5	4	2,29	2,5
	Satisfaction R	24	1,5	5	3,52	3	18	1,5	5	3,17	3
R19	Effectiveness ¬R	23	100	100	95,83	100	17	100	100	89,47	100
	Effectiveness R	24	100	100	100	100	18	100	100	100	100
	Efficiency ¬R	25	2,86	31,75	13,39	10,16	20	2,82	26,88	11,79	10,76
	Efficiency R	27	1,52	17,48	6,34	4,71	22	0,74	49,5	9,01	4,22
	Satisfaction ¬R	24	2,16	5	3,99	4	19	1	5	3,53	4
	Satisfaction R	24	1	5	4,26	4,42	18	2	5	4,01	3,92
R20	Effectiveness ¬R	17	100	100	70,83	100	15	50	100	76,32	100
	Effectiveness R	20	100	100	83,33	100	14	100	100	77,78	100
	Efficiency ¬R	20	0,53	26,67	5,93	1,02	18	0,2	21,6	6,49	2,46
	Efficiency R	23	0,23	13,19	2,85	1,41	18	0,3	6,92	2,31	1,28
	Satisfaction ¬R	24	1	5	3,29	3,5	19	1	5	3,37	4
	Satisfaction R	24	1	5	3,54	4	18	2	5	4,11	4
R21	Effectiveness ¬R	24	100	100	96	100	20	100	100	100	100
	Effectiveness R	25	100	100	96,15	100	18	100	100	85,71	100
	Efficiency ¬R	26	0,29	7,52	3,81	3,94	22	1,81	7,86	3,86	3,81
	Efficiency R	26	1,07	6,12	3,09	2,97	19	1,07	7,65	2,76	2,57
	Satisfaction ¬R	25	1,5	5	4,08	4	20	3	5	4,5	5
	Satisfaction R	26	3	5	4,54	5	21	3,5	5	4,55	4,5

APPENDIX C

Table C.1 Effort per refactoring

Nº	Refactoring	Easy	Medium	Hard
1	Improve the description of process links	X		
2	Split page		X	
3	Anticipate a validation activity		X	
4	Change widget	X		
5	Add a verification activity		X	
6	Add a "confirm and commit" activity	X		
7	Aggregate activities			X
8	Introduce information on demand		X	
9	Turn attribute into link		X	
10	Provide breadcrumb	X		
11	Move widget	X		
12	Split list		X	
13	Distribute menu			X
14	Keep the user informed on the ongoing process		X	
15	Improve link destination announcement		X	
16	Remove duplicated process links	X		
17	Make explicit the steps composing the process and the current step being executed		X	
18	Enable user to go back in the process steps			X
19	Add a summary activity		X	
20	Add processing page		X	
21	Add an "assistance" activity		X	