Electron Commer Res (2014) 14:497–529
DOI 10.1007/s10660-014-9149-0

Business processes refactoring to improve usability in E-commerce applications

Damiano Distante · Alejandra Garrido · Julia Camelier-Carvajal · Roxana Giandini · Gustavo Rossi

Published online: 25 September 2014

Abstract Refactoring is a technique that applies step-by-step transformations intended to improve the quality of software while preserving its behavior. It represents an essential activity in today's software lifecycle and a powerful tool against software decay. Software decay, however, is not only about code becoming legacy, but it is also about systems becoming less usable compared to competitor solutions adopting new designs and new technologies. If we narrow the focus on e-commerce systems, the role of usability becomes essential: higher usability is in fact a requirement to win the market competition and to retain customers from turning to other choices. One reason why an e-commerce application can start suffering from poor usability is because of its business processes (BPs) becoming difficult to access, complicated to execute, and, overall, offering a poor user experience. In this paper we argue that refactoring can be a key solution for this kind of usability issues. In particular, we propose a catalog of refactorings as a means to systematically identify and address lack of usability in the

D. Distante (✉)
Unitelma Sapienza University, Rome, Italy
e-mail: damiano.distante@unitelma.it

A. Garrido · J. Camelier-Carvajal · R. Giandini · G. Rossi
LIFIA, Fac. de Informática, Universidad Nacional de La Plata, La Plata, Argentina
e-mail: garrido@lifia.info.unlp.edu.ar

J. Camelier-Carvajal
e-mail: juliacamelier@gmail.com

R. Giandini
e-mail: giandini@lifia.info.unlp.edu.ar

G. Rossi
e-mail: gustavo@lifia.info.unlp.edu.ar

A. Garrido · G. Rossi
CONICET, La Plata, Argentina

BPs of an e-commerce application, and to seize opportunities for usability improvement. To make the presentation concrete and to provide evidence of the benefits that applying our refactorings can bring, we present a number of examples with reference to well-known e-commerce websites.

Keywords E-commerce websites · Business processes · Business Web applications · Usability · Quality-in-use · Refactoring · Web model refactoring · Web business process refactoring

1 Introduction

Usability can be defined as "*the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*" [24].

Several research works and market surveys have been conducted to measure the impact of website quality on trust, intention to shop, and consumer's commitment to an e-commerce website [8,20]. Other works have focused on identifying the usability factors which are considered more crucial for the website of a specific industry [30]. In addition, several studies have revealed the importance of usability for the success of an e-commerce website [2,19,34,45]. All these works motivate and foster the research on approaches and techniques for the evaluation and improvement of usability in e-commerce and business Web applications,[1] including the research presented in this paper.

One possible cause for the lack of usability in e-commerce websites is related to their business processes (BPs) turning complicated to access and execute, because of some maintenance activity, or for an improper initial design, which is the case when business goals are not well aligned with BPs, as argued in [42]. On the other hand, another critical reason to recommend usability improvements is the competitive pressure from other e-commerce sites that keep getting better [33]. In this paper we present a definition and a catalog of refactorings aimed at improving the usability of BPs when implemented by a Web applications

Refactoring was originally conceived as a disciplined technique for restructuring a class hierarchy of an object-oriented design [39]. Later, refactoring became popular and evolved into "a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior" [14]. Refactoring has been applied to different software artifacts, such as Unified Modeling Language [35] (UML) models [5], access control architectures [25], and HTML Web pages [22]. In all cases the basic philosophy of refactoring has been retained, i.e., each refactoring is a small, behavior-preserving transformation, aimed at improving some quality characteristic of a software artifact. Conversely, the set of quality characteristics, originally limited to internal quality attributes such as understandability and

[1] Here and elsewhere not differently specified in the paper we use the terms "website" and "Web application" interchangeably.

maintainability, has expanded to include external quality and quality-in-use attributes such as performance, security, usability, and accessibility.

In a recent work we have applied refactoring to the area of Web application design models by introducing the concept of web model refactoring (WMR): a small behavior-preserving change applied to the design models of a Web application, aimed at improving its usability [17]. We have also presented a catalogue of WMRs for the navigation and presentation models of a Web application, aimed at improving its usability [15,16]. Business processes have been already the focus of refactoring [52,53], but mainly to improve their internal quality features, and irrespective of their implementation in Web applications.

In this work we extend our research on WMRs to the realm of business processes and propose a catalogue of refactorings for the business processes implemented in e-business and e-commerce applications, still with the intent of improving their usability. Similarly to patterns catalogues, we illustrate refactorings with well-known uses, achieving with this a sound validation of them. Additionally, we show how to apply refactorings by identifying bad smells in these Web applications. Since our discussion is focused on the realization of BPs in Web applications, specifically on their navigation and user interface aspects, all references to business processes must be considered in that context, as many times, for the sake of readability, we omit to re-emphasize this.

The rest of the paper is organized as follows: Sect. 2 provides the background for our research work by identifying, on the one hand, similarities in a number of methods for designing business processes in Web applications, and on the other hand, the works on usability that inspired the refactoring catalog; Sect. 3 defines the concept of Web business process refactoring (WBPR) and provides a framework for their characterization; Sect. 4 introduces a representative set of WBPRs in the form of a catalogue; Sect. 5 presents some case studies and discusses some strengths and limitations of our approach for usability improvement; Sect. 6 surveys related work, and finally Sect. 7 concludes the paper and announces future work we aim to conduct.

2 Background

We briefly introduce two topics which are the basis of our research; first we explain how different Web engineering approaches deal with the problem of modeling BPs which will be implemented by a Web application, particularly, the different design concerns that are involved during the design activity. Next we reference seminal work on usability in general and in the Web.

2.1 The design of business processes in Web applications

Web applications have rapidly evolved over the past 10 years from content delivery and information provisioning websites (information-centric Web applications) to complex business Web applications supporting the realization of BPs, such as e-commerce websites. Because of their complexity and diversity, the development of e-business applications usually demands the use of a suitable Web engineering methodology which targets, among other aspects, the design and integration of BPs in Web applications [6,11,26,46,49]. In these methodologies, as a way to decouple different design concerns as discussed in [1], the design of a BP is generally accomplished at three different layers: *Process* layer, *Navigation* layer, and *Presentation* layer.

In particular:

- The Process layer specifies *structural* (or static) and *behavioral* (dynamic) characteristics of the BP, such as the set of activities it includes, their classification into user-driven and system-driven activities, their hierarchical composition in terms of activities and sub-activities, their dependencies (e.g., in terms of *data flow*) and semantic relations, the business rules which apply to their execution, the possible workflows among them (usually defined by means of activity *control-flows*), and the way different actors collaborate to the execution of the BP. Most Web engineering methods address these design concerns by relying on approaches for dealing with BPs in general software (see related work section).
- The Navigation layer defines how the execution of BP activities and the navigation through the Web application contents influence each other, how navigation may modify the state of an ongoing BP and the associated data, and which data is presented and/or requested to the user when executing a given activity.
- The Presentation layer describes the interaction widgets that allow triggering each BP activity, as well as the interface widgets that present/request data to/from the user for a given activity.

2.2 Usability in e-commerce applications

When defining our catalog of refactorings to improve the usability of Web applications, we were inspired by the work of giants who have researched on usability and web usability for many years, like Shneiderman [47] and Nielsen [31,32,34]. They have laid down the basis of what is "good" or "bad" in interface design, what "works" or "fails" to make the user comfortable with software. Particularly in this work, they provided us with principles, guidelines, and quality attributes that should be the targets of our refactorings, while also pointing to bad practices that we convey as "bad smells".

Some examples of how the works of experts have influenced the present work are:

- Shneiderman's "eight golden rules of interaction design" [47] includes general guidelines, applicable to any graphical interface, like "Permit easy reversal of actions". When adapted to the context of BPs in a Web application, a bad smell inspired by this golden rule is when a process does not allow to be canceled. A refactoring that solves this bad smell is "Make a process cancelable".
- Nielsen's books ([31,32]) and websites ([32,33]) provide several guidelines that although not directly related to BPs, motivate presentation refactorings, like "Make explicit the steps composing a process" motivated by Nielsen's discussion on navigation: "Where am I?" and "Where have I been?" [31].
- Lauesen work on virtual windows [28] advices us on the systematic design of interfaces, paying attention to the psychological laws of how users perceive what they see and do not see, under the basic idea that "important tasks need only a few windows". This basic but insightful idea motivates a number of our refactorings

including "Aggregate activities", "Make input data in one activity visible to other activities" (so to prevent repeatedly requesting input from the user), or "Remove duplicate process links" (so to reduce the screen space needed).

Cataloging these principles and guidelines in terms of a refactoring means giving a particular structure to the solution that has the following attributes: (*i*) can be performed in a sequence of small and safe steps, (*ii*) those steps can be repeated similarly in different applications and different contexts, (*iii*) the steps can be automated, and (*iv*) they can be applied on a working application without interfering on its existent functionality. These are all attributes of any refactoring, which are essential for today's business on the Web, to be able to create and maintain applications that are constantly changing in response to their users' requirements and needs [17].

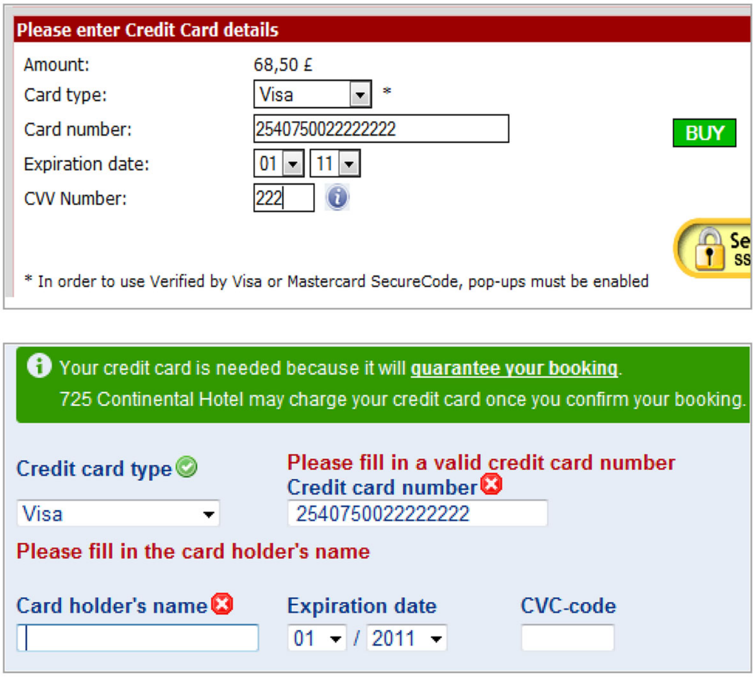
3 Defining and characterizing Web business process refactorings

The information stored in process definitions (e.g., task processing sequences and role information) can be leveraged to improve the UI of an existing e-commerce application, and thus its usability [54]. However, the fact that BPs exhibit themselves "just" as a sequence of activities does not guarantee their usability. As an example, users might get disoriented when navigating to pages outside a BP, or might get confused about how to complete a given BP activity or proceed to the next one. In complex e-commerce applications involving BPs, such as checking-out in an e-store or booking a flight or hotel in a online reservation system, users spend a lot of time in process activities such as forms filling and options selection (e.g., for user registration, order shipping and billing address specification, etc.), data verifications (e.g., credit card verification), transactions execution (e.g., credit card charge), and process completion confirmation. When these process activities and/or their control-flow are not wisely designed and implemented, the usability of the overall Web application may be seriously compromised. For instance, an improper BP design or implementation might cause the user getting disoriented by a cluttered interface, or uselessly repeating some step of the process (e.g., entering personal data), or leaving a process for not receiving any feedback from a long lasting transaction.

As an example, we show in Fig. 1 two different implementations of a process that requests and verifies credit card data for online hotel reservations. The screenshot on the top is from www.hotelbooking.com. Here the user enters the credit card data and, after clicking the "Buy" button, data is verified. If the user misspells her credit card number, she has to re-enter all her data again. The screenshot on the bottom is from www.booking.com. In this case, verification occurs while the user fills each field, so she does not have to restart the process if any of the fields is misspelled.

This is an example of the kind of usability enhancements that we seek. Other examples are: attaining a better process workflow, a better allocation of content in the screen [31] or among available pages, and a better support for the user while executing the BP.

We consider the kind of usability enhancement that we have just illustrated a *refactoring*, in the sense that we have defined in [17]: a change applied on a Web application



Please enter Credit Card details

Amount:	68,50 £
Card type:	Visa *
Card number:	2540750022222222
Expiration date:	01 / 11
CVV Number:	222

BUY

* In order to use Verified by Visa or Mastercard SecureCode, pop-ups must be enabled

Your credit card is needed because it will guarantee your booking.
725 Continental Hotel may charge your credit card once you confirm your booking.

Credit card type: Visa

Please fill in a valid credit card number
Credit card number: 2540750022222222

Please fill in the card holder's name

Card holder's name:

Expiration date: 01 / 2011

CVC-code:

Fig. 1 Two different implementations of the credit card data input and verification activity in a hotel booking process. The first screenshot is from www.hotelbooking.com, while the latter is from www.booking.com

BP with the intent of improving its usability and which preserves the expected functionality and result. While there are other definitions of business process refactorings [52], they focus mainly on the organization of BP activities mainly to improve internal quality properties (e.g. model readability, modularity, ease of maintenance, etc.). In this sense they have the same intent of the original Fowler's refactorings [14], and thus they do not have any impact over the application's usability. Though we comment these other approaches in detail in the related work section, it is important to stress here that none of them overlaps our refactorings, even partially. The reason is that in this paper we are interested in those refactorings which (a) are meaningful in the Web interface of a BP and (b) are intended to improve the application's usability. Therefore our refactorings are novel not just in their intent but also in their scope, structure, and mechanics.

To make our definitions more precise we assume that a business process P encompasses a set of activities A; some of these activities are system-driven, i.e. they don't need any user intervention to be completed, while others are user-driven since they require some user intervention, e.g. to fill a form, confirm some options, etc. In our research we are interested in those activities $A' \subseteq A$ which, being user-driven, have a Web interface. This interface (usually a Web page) will exhibit contents related with the BP itself and with the corresponding activity $A_i \in A'$, links to other pages, activity-related forms, interface widgets to control the flow of process P, etc. As expressed below, our refactorings are concerned with this kind of activities and their Web interfaces.

More precisely, we define a *Web Business Process Refactoring* (WBPR) (or usability refactoring for the BPs of a Web application) as a change applied to a BP implemented by a Web application, which has the following properties:

- It is perceived by the final user of the application, i.e., it is applied to a user-driven activity;[2]
- It is intended to improve the application user's experience in reaching the goals underlying the BP itself;
- It preserves the set of use cases and requirements associated to the BP that the application satisfies and that can be checked against acceptance tests, it does not modify any business rule, nor add to the process any new system-driven behavior, i.e., a behavior unrelated to the user interface and its enhancement. In other words, a WBPR cannot break any user acceptance test that applies to the Web application.

Under the above conditions and specifically regarding user-driven activities, a WBPR may:

- Group or split activities and the associated Web interfaces.
- Modify the process control-flow (e.g., by moving or copying an activity in a process, or by parallelizing two non-dependent activities).
- Make an activity be optional, suspendable, or cancelable.
- Add a support activity, i.e., an activity that clarifies a process or makes it more effective, but does not change its functionality.
- Improve the navigation associated to the execution of a process (e.g., by extending the available navigation structure or removing redundant links).
- Improve the presentation of a BP as a whole (e.g., by making explicit its execution state), and that of its activities and their related data (e.g., by changing the widget used to execute an activity, or to present/request data associated to it).

Regarding acceptance tests, we note that in the same way as with traditional refactoring, the better the test suite that the application has, the more confidence it gives to designers and developers to try refactorings without breaking the application's functionality. In the case of automatic refactorings, a refactoring tool can compensate for missing tests by checking preconditions [14] or checking the preservation of flow dependencies [44]. In particular, WBPRs should preserve *data flow* dependencies between BP activities and, more generally, the state of the workflow variables when these variables need to be used [29]. The approach of Weber et al. [52] considers refactorings as model transformations that preserve "*execution trace equivalence*" between process models. We believe that preserving a BP's execution trace or control flow is too restrictive and does not allow for substantial improvements on usability and other quality in use characteristics of Web applications.

Notwithstanding, in this article we are not interested in providing a formal specification of behavior preservation of WBPRs for two main reasons: (i) the specification would depend on a particular programming language or Web design methodology

[2] But not necessarily, or not only, to the Web interface of such user activity.

while our purpose is to present a high-level view that may apply at the model or code level in any language, and (*ii*) we are interested in emphasizing the benefits of using WBPRs through an initial catalog of refactorings and encourage practitioners and researchers to augment it, as opposed to restrain them.

A WBPR may improve the user's experience concerning quality in use characteristics of the application such as effectiveness, efficiency, and satisfaction. Moreover, since BPs are designed in a Web application at three different layers (i.e., process, navigation, and presentation), a WBPR may imply changes to any of the corresponding design layers of the application. By naming *Intent* the set of quality characteristics a WBPR aims to improve and *Scope* the BP design layers and artifacts the WBPR impacts, we developed a simple characterization framework for WBPRs presented in Sect. 4.

To classify the Intent of a WBMR we refer to the quality-in-use characteristics defined by the Quality in Use Model of the ISO/IEC 25010 standard for system and software [24], which represents a broader view of the ergonomic concept of usability defined in the ISO 9241-11 standard [4], [23]. The ISO/IEC 25010 standard defines *Quality in Use* as, "*the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use*". A brief explanation of these keywords in usability follows:

- *Effectiveness*: accuracy and completeness with which users achieve specified goals.
- *Efficiency*: resources expended in relation to the accuracy and completeness with which users achieve goals.
- *Satisfaction*: degree to which user needs are satisfied in a specified context of use, including: Usefulness, Trust, Pleasure, and Comfort.
- *Freedom from risk*: degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment.
- *Context coverage*: degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and in contexts beyond those initially explicitly identified.

4 A catalogue of business process refactorings for e-commerce applications

Following the definition provided in Sect. 3, we have developed a catalogue of WBPRs aimed at improving e-commerce applications' quality-in-use characteristics associated to the execution of their BPs. In this section we describe each of the refactorings in the catalog by motivating them with practical examples and describing them with a pattern-like template which comprises *Intent, Bad smell, Motivation, and Example.* As described in Sect. 3, the *Intent* is defined according to the Quality in Use Model of the ISO/IEC 25010 standard for system and software [24]; *Bad smell* is the indicator of the lack of usability or of the usability improvement opportunity that may suggest applying the refactoring; *Motivation* describes the problem that causes the bad smell; *Example* describes the application of the refactoring on a concrete example. The *Scope* of a refactoring, i.e., the design concern in which it applies, is implicit in our organization in categories (see the following of this Section).

To emphasize that our WBPRs may be applied on different models (UML activity diagrams, BPMN [36]), or could even be realized on Ambler's storyboards [3], we include the *Mechanics* (i.e., the steps to apply the refactoring), though, for the sake of conciseness, we do so only in the first refactoring of each category.

While each refactoring is in principle applicable to any Web application independently of the design methodology and implementation technologies adopted to develop the application, it is a task of the designers or developers of the specific BP and Web application to check that applying the specific refactoring not only preserves data flow dependencies, but also does not break any of the business requirements available for the application.

The list of refactorings that follows is organized into subsections depending on the Web design layer each refactoring mainly impacts. In turn, refactorings on the process design layer are divided into structural changes and behavioral changes. It is worth noting that refactorings introduced at the Process layer usually require changes also at the Navigation and Presentation layers, while the opposite usually does not happen, i.e., presentation WBPRs do not require changes in the Navigation and Process layers.

A summary of our catalogue of WBPRs with their characterization in terms of intent, scope and quality attributes they aim to improve is presented in Table 1.

4.1 Refactorings for the process design layer

Structural WBPRs (changes to the hierarchical and semantic relations between BP activities)

S1) *Aggregate activities*
Intent: Effectiveness and Efficiency.
Bad Smell: Long time to complete a simple task.
Motivation: Simple activities may be aggregated into a single one with the purpose of: reducing the interaction between client (the browser) and server, making it less cumbersome for the user to enter just a few data items in different pages or at different stages, and also to expedite a process by aggregating the underlying activities.
Example: In order to access the home banking system of "Banco de la Nacion Argentina" www.bna.com.ar as well that of "Banca Monte dei Paschi di Siena" www.mps.it, the login page requests only the username first, and, after a validation in the server, the password input field is requested. Both activities are very simple (with a single input field in each one) and being separated requires an interaction between the server and the client that increases the time to complete the task. Instead, both input fields may appear at the same time reducing the steps and accelerating the process completion. This happens for "Banca Intesa" www.bancaintesa.it as well for "BNL—Group BNP Paribas" www.bnl.it.
Mechanics: In Fig. 2 we use the notation of the UWE methodology [26] to describe the process flow for the login operation in the BNA website. With the username and password displayed as separated activities, the user has to wait the system to process and validate both in two different times, as shown in Fig. 2 on the left. As suggested in this refactoring, grouping both activities in one, as shown in Fig. 2

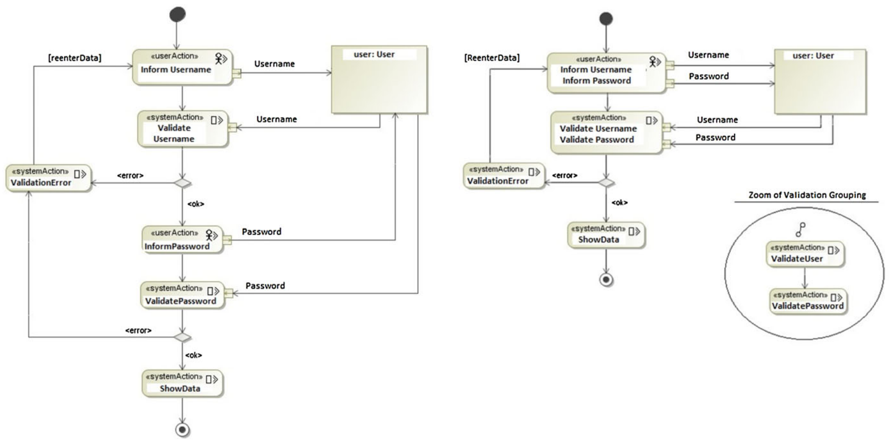


Fig. 2 UWE's model showing the process flow for the user login operation

(right), makes the system also to group its validations, thus causing less effort and waiting time to the user, that only has to press one button and wait once for the validation.

S2) *Split activity*
Intent: Effectiveness, Satisfaction, Freedom from Risk and Context Coverage.
Bad Smell: Activity too long; premature abandonment of the application.
Motivation: A complex activity may be split into smaller ones for different reasons: to emphasize some portion of the original activity; to enable the user suspending and resuming it in/from an intermediate completion state; to adapt the application user interface to mobile devices, etc.
Example: Companies that allow prospective employees to enter their curriculum vitæ on their websites should have the activities split into different pages, as Personal Information, Work Experience, etc. Since this is a complex task that requires some time to be completed, splitting it makes the activity more organized to the user and allows applying a subsequent refactoring to make it suspendable.

S3) *Change an activity from mandatory to optional and vice versa*
Intent: Efficiency and Satisfaction.
Bad Smell: Unnecessary activities in the main process.
Motivation: An activity may be changed from "required" to "optional" in order to offer a shorter process workflow.
Example: The activity of choosing a seat or specifying the desired type of meal on a flight could be changed from mandatory to optional in order to expedite the reservation process.

S4) *Change dependencies between two activities*
Intent: Satisfaction.
Bad Smell: Bad hierarchy in the process: Business rules not respected.
Motivation: The completion of an activity may be changed from optional to required in order for another activity to be started, or for an enclosing activity

to be completed, when this increases safety, enforces business rules, or when it improves the organization and understandability of the steps of a process.
Example: For example, in an e-learning system, the completion of a learning unit may be defined as "required" in order for the following one to be started, or passing an assessment may be changed into required in order for the learning unit to be considered completed.

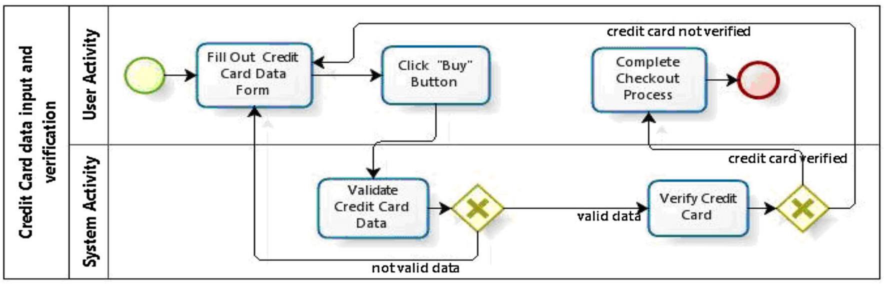
S5) *Make input data in one activity visible to other activities/processes*
Intent: Effectiveness and Efficiency.
Bad Smell: Duplication of steps within the process.
Motivation: Instead of requesting users to enter the same information in more than one place, transfer the information directly to the processes/activities that require it after it has been entered once.
Example: At Amazon.com, the sign-in activity required for a user to access her personal data or previous orders information is not visible to the checkout process, so that the user is requested to login again to complete an order.

S6) *Make an activity suspendable*
Intent: Efficiency and Satisfaction.
Bad Smell: Premature abandonment of the application; long time to conclude a process.
Motivation: Making an activity suspendable enables the user to pause the process during the execution of that activity and restart it afterwards, from the same point, possibly using a different device. This may result in an improved efficiency and a better user satisfaction, especially when a given activity is particularly complex and requires time to be completed.
Example: The option to save a post in a blog server (i.e., Blogger, Wordpress, etc.) allows the user to start an activity, suspend it and resume it later, providing the possibility to review the post as many times as necessary before publishing it.

Behavioral WBPRs (changes to the BP control flow)

B1) *Change the order of execution of two or more activities*

B1.a) *Anticipate a validation activity*
Intent: Effectiveness, Efficiency, Satisfaction and Context Coverage.
Bad Smell: Users repeatedly filling a form because of subsequent failed validation.
Motivation: The system activity that verifies all data in an input form after it has been submitted could be anticipated and triggered as the user fills each of the form fields, thus to improve efficiency and effectiveness, since the user will not wait to receive feedback when entering erroneous or incomplete data.
Example: An example of the above bad smell appeared back in Fig. 1 at the top, which shows the process of requesting and verifying credit card data for reservations at www.hotelbooking.com.
Mechanics: We show this refactoring on a BPMN model. Figure 3-top shows the BPMN model of a "Credit Card Data Input and Verification" process in www.booking.com. The activity "Validate Credit Card Data" runs after the "Fill Out Credit Card Data Form" activity is completed and the "Buy" button is clicked. If the validation fails, the systems requests the user to re-enter the data



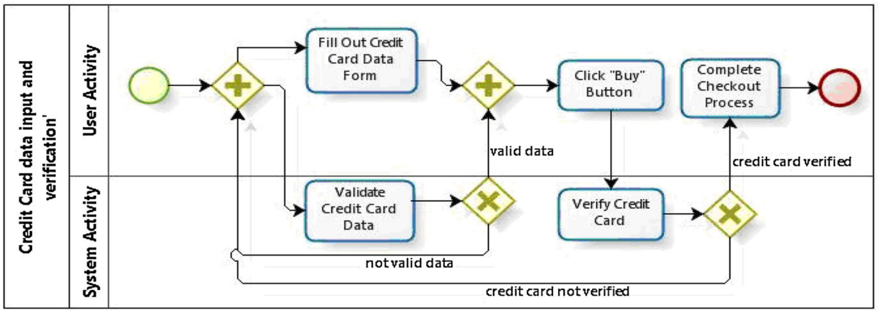


Fig. 3 The process of Credit Card Data Input and Verification (*top*) and its refactored version (*bottom*) obtained by anticipating the Validate Credit Card Data system activity

and the sequence is repeated. The activity node "Validate Credit Card Data" can be anticipated by inserting a parallel gateway to access either the selected activity or the previous one. In the model of Fig. 3-bottom, obtained after applying the Anticipate Activity refactoring, the parallel gateway specifies that "Fill Out Credit Card Data Form" and "Validate Credit Card Data" will run in parallel. Then, if the validation is successful, the flows are joined together to continue the process execution.

B1.b) *Postpone the execution of an activity*
Intent: Effectiveness, Efficiency, Satisfaction and Context Coverage.
Bad Smell: Premature abandonment of the application.
Motivation: An activity that is not required to complete a portion of a process in which there is a loop should be postponed after the loop, in order to avoid uselessly repeating its execution. This would avoid user discontentment to go through a non-essential activity several times.
Example: In the process of booking a flight, the activity of choosing a seat should be postponed with regard to the sub-process of searching for a flight, which is usually repeated several times until the user finds a suitable flight. This was a problem present in a previous version of the www.alitalia.com website (and documented in [48]) that has been corrected.

B1.c) *Make two activities executable in parallel and/or without a specific order*
Intent: Effectiveness, Efficiency, Satisfaction and Context Coverage.

Bad Smell: Process Inflexibility.

Motivation: Enabling two or more activities of a process to be executed in parallel when no dependency exists between them may provide the opportunity for two users to collaborate in a process and speed-up its execution.

Example: As an example, in an e-procurement system, some of the activities of preparing a competitive tender can be parallelized and executed by more administrative officers in parallel. At the same time, a given user may execute some activities which are of support to a given activity in parallel with it. Under the same conditions of data and control flow independence, enabling two or more activities to be executed with no predetermined order makes the process more flexible and efficient. In this case, in fact, the user may choose the activity to execute depending, for example, on the currently available business data or for opportunity reasons, and proceed with the others as data they require are made ready. Conversely, a strictly sequential workflow, when not required might reduce process flexibility and user freedom.

B2) *Add a support activity*

B2.a) *Add an "assistance" activity*

Intent: Effectiveness, Efficiency, Satisfaction and Freedom from Risk.

Bad Smell: Frequent empty results because of erroneous data in search fields; many backward-link activation to correct form data.

Motivation: When the user can enter free (and possibly erroneous) data in an input form, and the set of possible (and correct) input data is limited, it is possible to reduce the errors in input data to increase efficiency and satisfaction in using the application.

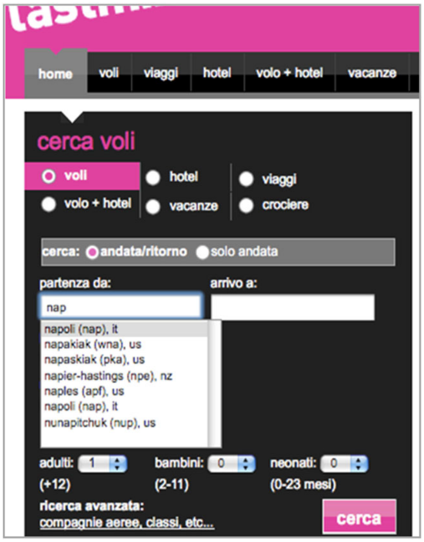
Example: An example of such activity is the autocomplete feature that helps the user in filling in the fields of a form faster and free of errors. This decreases the chances to have empty results or repetition in the process due to incorrect data entered by the user. In Sect. 5 we show the screenshots of www.lastminute.com showing this bad smell while entering data to search for a flight. Figure 4 shows the Italian version of the site, which thanks to the use of the autocomplete feature, the user is smoothly prompted with a list of airports corresponding to the initials she has inserted. This speeds up the form filling process and reduces the chances for erroneous input.

Mechanics: We show this refactoring over a UML Activity Diagram. In this case, every user activity to enter data is parallelized with a new activity that auto-completes form fields, either filtering the possible answer set or recovering from saved customer data.

Figure 5 shows on the left the UML activity diagram of the process of searching for a flight at www.lastminute.com and on the right its refactored version obtained by adding the autocomplete support activity "Provide Data Input Support by Form Fields Autocomplete", which is already available in the Italian version of the site.

B2.b) *Add a verification activity*

Intent: Effectiveness, Efficiency, Satisfaction and Freedom from Risk.



voli · hotel · viaggi · volo + hotel · vacanze · crociere

cerca: andata/ritorno · solo andata

partenza da: nap

arrivo a:

napoli (nap), it
napakiak (wna), us
napaskiak (pka), us
napier-hastings (npe), nz
naples (apf), us
napoli (nap), it
nunapitchuk (nup), us

adulti: 1 (+12) bambini: 0 (2-11) neonati: 0 (0-23 mesi)

ricerca avanzata: compagnie aeree, classi, etc...

Fig. 4 Autocomplete support activity in the page for searching for flights at www.lastminute.it

Bad Smell: Users feeling uneasy to enter personal data. Repeated phishing or bot automated attacks.
Motivation: Phishing and security threats are rising every day. Users need to feel secure to enter personal data or credit card information, otherwise they will abandon the application [31]. A small verification activity (like replying to an email) can make a difference to build trust in those users aware of security threats, and at the same time can ensure website security in the case of users unconscious of security warnings.
Example: A CAPTCHA test is a short activity to which users are getting accustom, intended to verify that a request to a website originates from humans, and also as anti-phishing. It may be introduced just before the starting activity of a public accessible process, or before the last activity completing it.

B2.c) *Add a summary activity*
Intent: Effectiveness, Efficiency, Satisfaction and Freedom from Risk.
Bad Smell: Risk of errors.
Motivation: In the context of a purchase process, for example, it is very useful to add an activity to enable the user verifying the content of her shopping cart and modify the quantity of each product before proceeding with the checkout.
Example: In www.amazon.com, the user is able to review the order before actually paying for it. He can check the quantity and items purchased, address that it's going to be delivery, details of the payment. This way he can be that everything is correct before committing to buy the items.

B2.d) *Add a "confirm and commit" activity*
Intent: Effectiveness, Efficiency, Satisfaction, and Freedom from Risk.
Bad Smell: Hesitation to move forward in the process execution towards its completion.

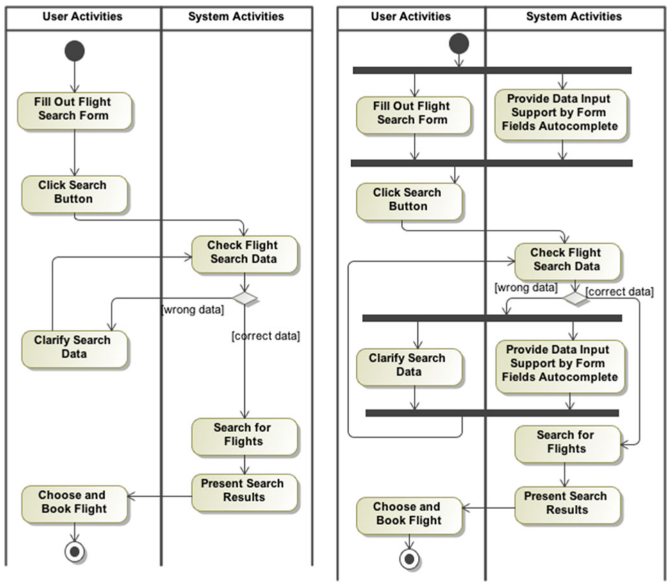


Fig. 5 The process of searching for a flight at www.lastminute.com (*left*) and at lastminute.it (*right*), with the latter featuring the "Provide Data Input Suggestions" support activity

Motivation: Adding a "confirm and commit" activity as the last step before completing a process and committing the associated transaction may increase system trustability. Advertising the availability of such an activity at the end of a process may help the user trusting the system and executing the process activities without worries.

Example: In well designed check-out processes, such as in www.amazon.com, after the user has provided all needed data, there is a final confirmation activity (sometimes blended with the refactoring "Add a summary activity" described in B2.c).

B2.e) *Make a process cancelable*

Intent: Efficiency, Satisfaction, and Freedom from Risk.

Bad Smell: Reduced process flexibility; inconsistency generated by the use of the browser forward and back buttons.

Motivation: Making a process cancelable whenever possible during its execution by explicitly offering a "Cancel" activity may increase trust and improve user satisfaction.

Example: The current design of the checkout process at www.amazon.com misses offering such a feature to the user and the only way for her to cancel a started process is to go back in the browser history or to directly type a new URL in the browser address bar, or even closing the browser. The first behavior may originate an incoherent state in the ongoing process, while the second and third ones are very frustrating for the user.

4.2 Refactoring for the navigation layer

N1) *Reduce the number of navigation links provided to the user while executing a process*
Intent: Effectiveness, Efficiency, Satisfaction, and Freedom from Risk.
Bed Smell: Possible inconsistencies in the ongoing process; user distraction.
Motivation: Having too many navigation links displayed while executing a BP which are not necessary to accomplish the BP or which do not observe any specific grouping and layout criteria, may be confusing and distracting to the user. This refactoring is intended to reduce or remove such links, so that the user may focus on the activities required to be executed to complete the process. It also avoids an inconsistency of the process in case the user decides to click in a link that doesn't belong to the BP.
Example: In the UFV online bookshop (www.editoraufv.com.br) during the checkout process, the user has access to many navigation links that might obscure the process, breaking the process flow with navigation operations.
Mechanics: The UWE navigation diagram [26] in Fig. 6-top has been simplified in Fig. 6-bottom, by eliminating most links, except those pertaining to the checkout BP.
It is important to remark that designers might choose intermediate solutions to similar examples; for instance we could maintain the link to the Products navigation class, therefore allowing the user to check for similar products (or even the ones he chose during the buying process).

N2) *Keep the user up to date on the ongoing process*
Intent: Effectiveness, Efficiency, Satisfaction, Freedom from Risk, and Context Coverage.
Bad Smell: Lack of information about the current process; unnecessary repetition of steps.
Motivation: Keeping the user informed of the current status of an ongoing process is another way to increment trust and satisfaction.
Example: In an online store, in order to keep the user informed of the ongoing shopping process, the current status of her shopping cart (i.e., the list of products it includes and their total cost) can be shown in a sidebar presented in every page of the site while browsing it for products to buy and until the checkout process is started. Figure 7 shows, the demo store of the Magento e-commerce software platform www.magento.com.
Differently from Amazon, the Magento e-commerce solution features a shopping cart sidebar showing the list of products currently included in the shopping cart and their total cost. Having the status of the shopping cart at hand also reduces the need for the user to navigate to the shopping cart page in order to check its content and makes the shopping process more smooth and efficient. We show these differences in Sect. 5.

N3) *Improve the information provided to the user while executing an activity*

N3.a) *Improve the description of process links*
Intent: Effectiveness, Satisfaction, and Freedom from Risk.
Bad Smell: User confusion.

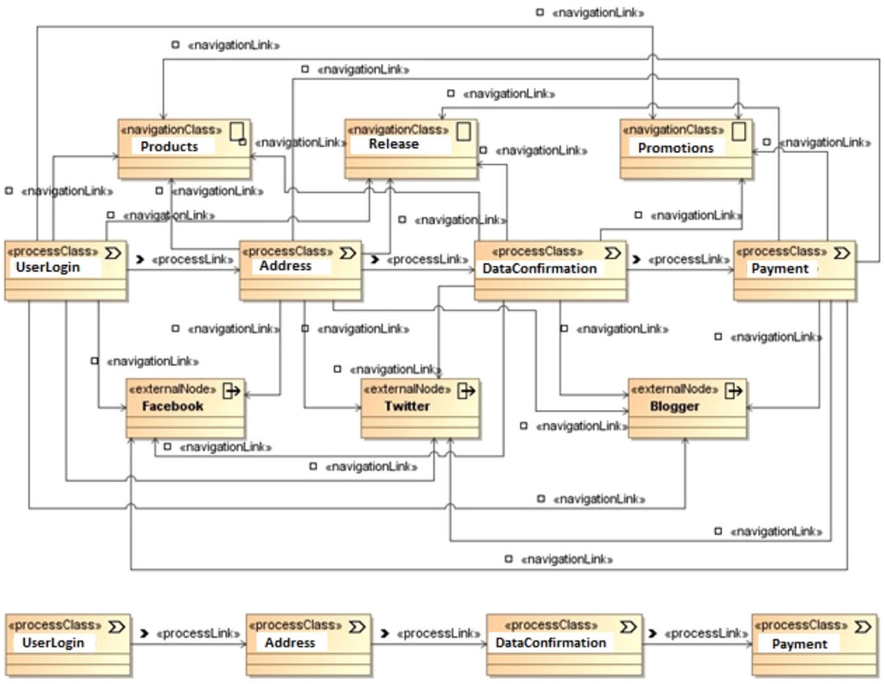


Fig. 6 Navigation Links showed with the BP page (*top*) and its refactored version (*bottom*)

Motivation: Process links well described can communicate their intent and avoid the user to erroneously click on them.
Example: In the www.amazon.com website, the hyperlink "Not <username>?" shown on top of each page and that has the effect to log off the current user, could be changed into something more explicit such as the text "Sign out".

N3.b) *Clearly describe errors in executed activities*
Intent: Effectiveness, Satisfaction, and Freedom from Risk.
Bad Smell: Lack of information about the process; user confusion; unnecessary repetition of steps.
Motivation: Precisely describing errors in input data is very important to avoid customer frustration. It can also refrain the user to repeat a step due to incorrect data entering.
Example: For example, when the user fills out a form with wrong data, the application should clearly indicate which data item is wrong and why.

4.3 Refactoring for the presentation layer

P1) *Make explicit the steps composing a process and the current step being executed*
Intent: Effectiveness and Satisfaction.
Bad Smell: Lack of information about the ongoing process; premature abandonment of the application.

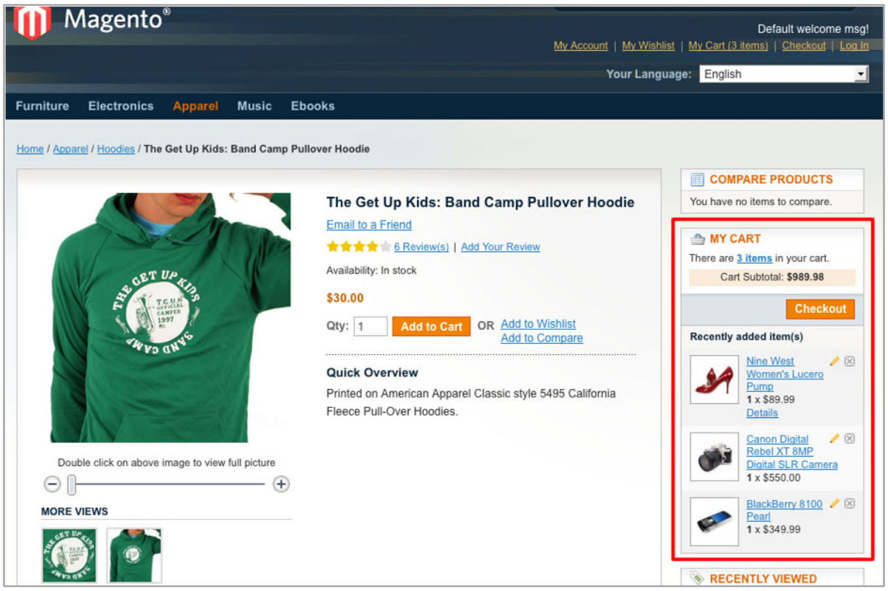
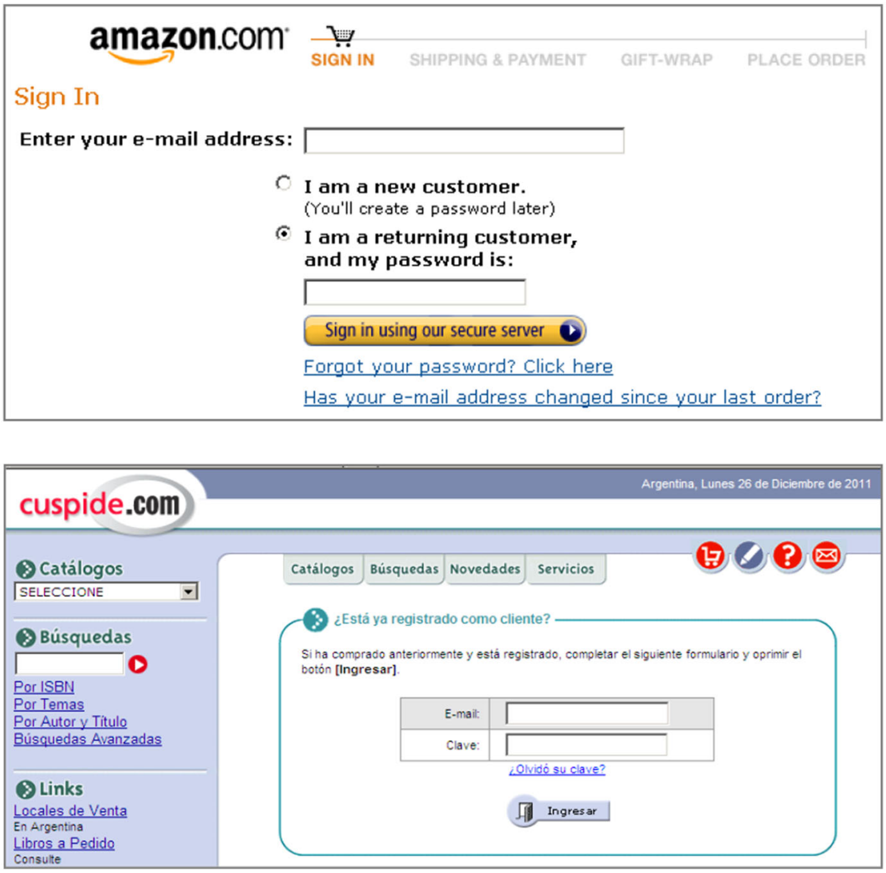

Magento
Default welcome msg!
My Account | My Wishlist | My Cart (3 items) | Checkout | Log In
Your Language: English
Furniture Electronics Apparel Music Ebooks
Home / Apparel / Hoodies / The Get Up Kids: Band Camp Pullover Hoodie
The Get Up Kids: Band Camp Pullover Hoodie
Email to a Friend
6 Review(s) | Add Your Review
Availability: In stock
$30.00
Qty: 1 Add to Cart OR Add to Wishlist Add to Compare
Quick Overview
Printed on American Apparel Classic style 5495 California Fleece Pull-Over Hoodies.
Double click on above image to view full picture
MORE VIEWS
COMPARE PRODUCTS
You have no items to compare.
MY CART
There are 3 items in your cart.
Cart Subtotal: $989.98
Checkout
Recently added item(s)
Nine West Women's Lucero Pump 1 x $89.99 Details
Canon Digital Rebel XT 8MP Digital SLR Camera 1 x $550.00
BlackBerry 8100 Pearl 1 x $349.99
RECENTLY VIEWED

Fig. 7 The page presenting a product in the Magento demo store (www.demo.magentocommerce.com) featuring a *sidebar* (highlighted in *red*) showing the current status of the shopping cart, with included items and total cost. (color figure online)

Motivation: Users requesting to execute a process in a website find valuable to know in advance the set of activities composing the process, how long the process will take and, during the process execution, which is the current activity being executed. For example, when filling out online surveys, which are usually requested to customers or to selected crowds without reward, it is important to describe the different sections of the survey and the average time needed to complete it, as a means of showing care for participants' precious time. This can be accomplished by showing a process status bar which: (*i*) lists the set of activities composing the process, (*ii*) highlights the current activity, and (*iii*) distinguishes already completed activities from those yet to be executed.

Example: Figure 8-bottom reports a page from www.amazon.com that presents on top of it the status bar for the process of checkout. The bar clearly indicates that the process is structured into four steps, "Login", "Shipping & Payment", "Gift-Wrap", "Place Order", and that "Login" is the current activity being executed. Such process status bar is instead missing in the www.cuspide.com e-bookstore (Fig. 8-top).

Mechanics: In this case we use the presentation model of the UWE methodology. Figure 9 shows the UWE presentation model representing the www.cuspide.com checkout page before (left) and after (right) applying the refactoring. The model on the left represents the current version of the page (screenshot reported in Fig. 8-top). In the model on the right, the process status bar was added as a Presentation Group that will contain the widgets that will inform the user about the step she is at, the steps already passed, and the ones to come until the process ends.


amazon.com

SIGN IN SHIPPING & PAYMENT GIFT-WRAP PLACE ORDER

Sign In

Enter your e-mail address:

I am a new customer.
(You'll create a password later)

[✓ I am a returning customer, and my password is:]

Sign in using our secure server

Forgot your password? Click here

Has your e-mail address changed since your last order?

cuspide.com

Argentina, Lunes 26 de Diciembre de 2011

Catálogos
SELECCIONE

Búsquedas
Por ISBN
Por Temas
Por Autor y Título
Búsquedas Avanzadas

Links
Locales de Venta
En Argentina
Libros a Pedido
Consulte

Catálogos Búsquedas Novedades Servicios

¿Está ya registrado como cliente?

Si ha comprado anteriormente y está registrado, completar el siguiente formulario y oprimir el botón **[Ingresar]**.

E-mail:	
Clave:	

¿Olvidó su clave?

Ingresar

Fig. 8 The page for starting a checkout process at www.cuspide.com (*top*) and at www.amazon.com (*bottom*), with the latter featuring a process status bar on the *top* of the page

P2) *Change the widget used to execute an activity*
Intent: Efficiency, Satisfaction and Freedom from Risk.
Bad Smell: Risk of error.
Motivation: An appropriated widget can make the user execute faster simple activities of the process and prevent possible errors. Simple text anchors may be replaced by buttons for better clarity; cursors are better than free textbox to specify values in a range; JavaScript calendar widgets are better than dropdown-lists to specify dates, etc.
Example: Figure 10 compares the pages for searching for a flight in the American and Italian versions of the www.lastminute.com website: additionally to dropdown-lists, the latter also offers a JavaScript calendar widget which makes specifying the flight departure and return dates fast and error free.

P3) *Remove duplicated process links*
Intent: Effectiveness, Satisfaction, and Freedom from Risk.

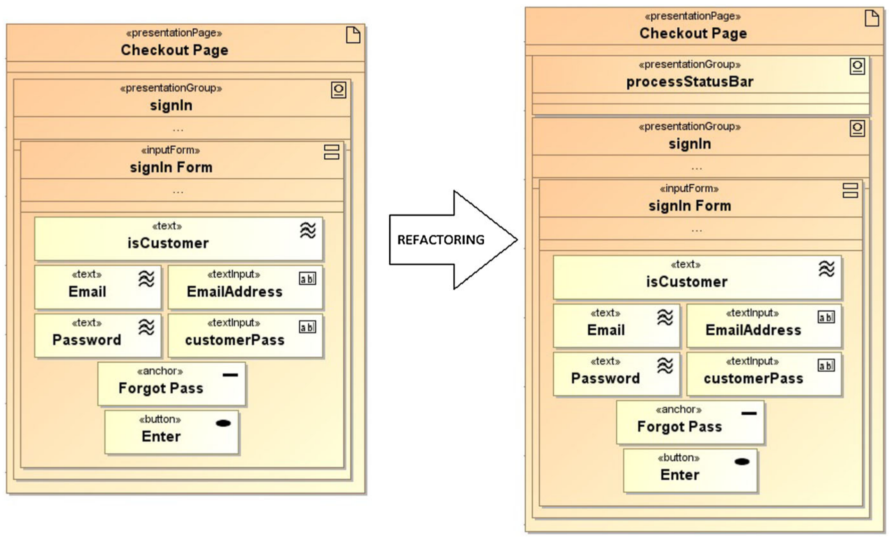


Fig. 9 The UWE presentation model representing the checkout page at www.cuspide.com, before (*left*) and after (*right*) applying the Add a Process Status *Bar* refactoring

Bad Smell: User confusion; redundancy within the BP.
Motivation: Sometimes a process link that leads the user to a given action appears in a Web page more than once, creating a redundancy that might confuse the user by giving the idea that they represent different things.

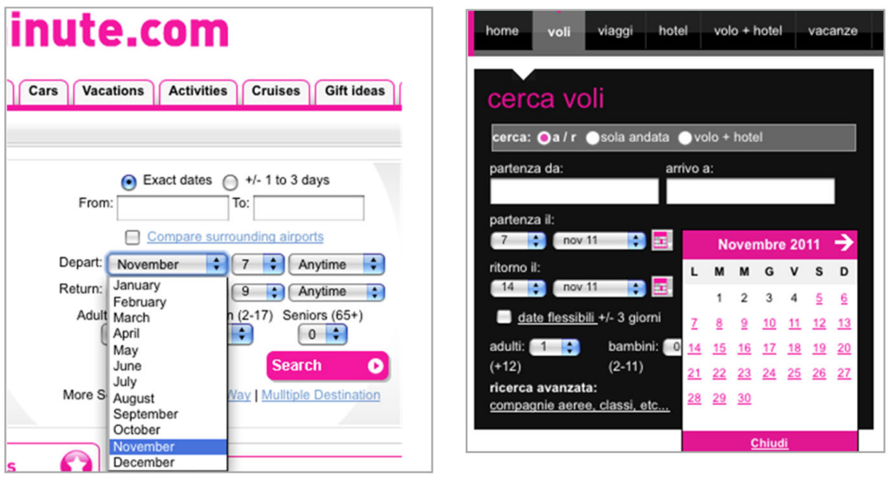
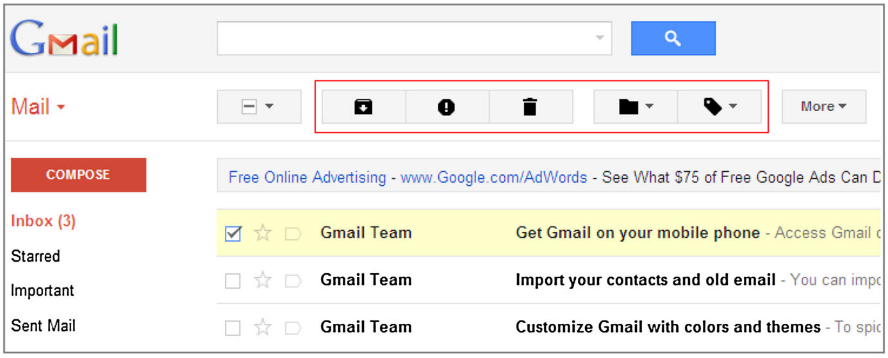


Fig. 10 The page of searching for a flight at www.lastminute.com (*left*) and www.lastminute.it (*right*), with the latter featuring a JavaScript calendar widget for selecting the departure and return dates



Free Online Advertising - www.Google.com/AdWords - See What $75 of Free Google Ads Can D

Gmail Team Get Gmail on your mobile phone - Access Gmail c

Gmail Team Import your contacts and old email - You can impc

Gmail Team Customize Gmail with colors and themes - To spic

Fig. 11 Example of process links that apply on a selected e-mail message in Gmail. All process links (*red* marked) are grouped and have the same look and feel. (Color figure online)

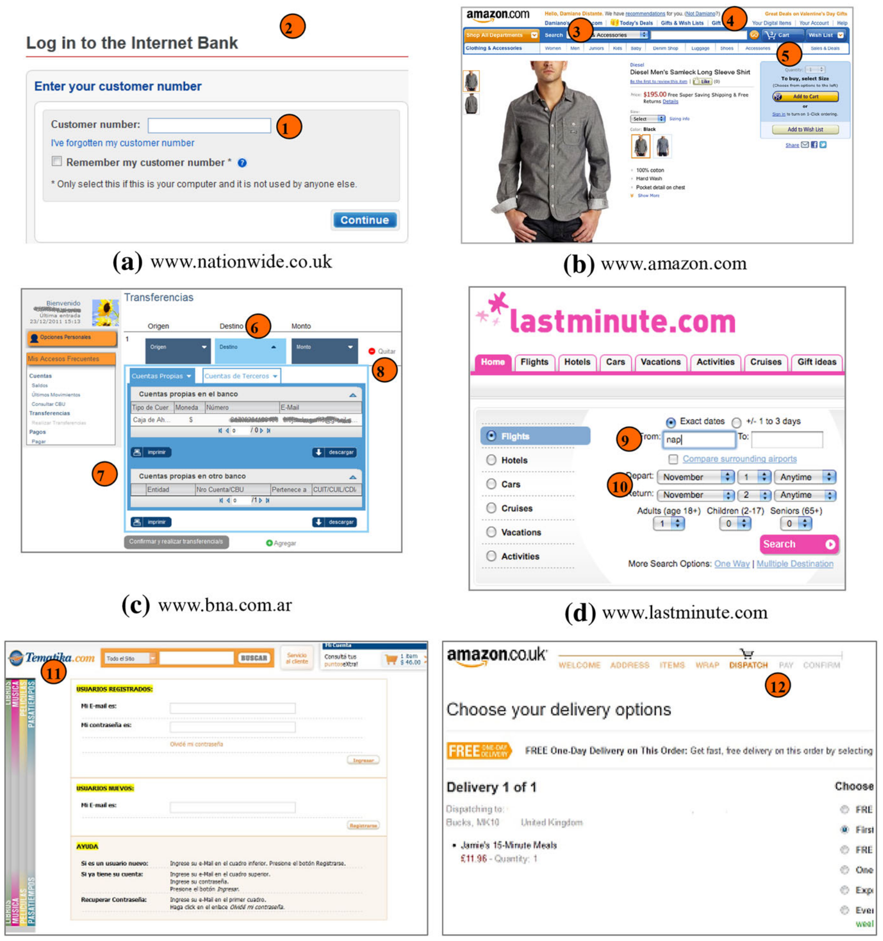
Example: In Fig. 12-c we can see two "Print" buttons that print exactly the same information: the bank account selected to receive the money transfer. Using this refactoring, the duplicated "Print" button should be removed to make the Web application more consistent and easier to understand.

P4) *Group process links that operate on the same domain entity*
Intent: Effectiveness, Efficiency, Satisfaction, and Freedom from Risk.
Bad Smell: User confusion.
Motivation: The way process links are distributed along a Web page as well as their look-and-feel help the user understand and identify groups of actions operating on the same domain entity. When process links are not grouped and have a different look-and-feel, it is hard to tell on which entity each one applies, thus generating confusion.
Example: In the Banco Nacion Argentina website, process links shown in the page for money transfer are much disorganized as shown in Fig. 12-c. Links such as "Quitar" (Delete) and "Agregar" (Add), which appear on opposite corners, refer to the same domain entity: a transfer. Moreover, "Confirmar y realizar transferencia/s" (Confirm transfer) has a different look-and-feel but also operates on the transfer. On the contrary, in the Gmail application (www.gmail.com) (Fig. 11), when the user selects an email in the Inbox, some buttons/links such as, "Delete", "Reply", "Reply All", "Move", etc., appear all together and with the same look-and-feel. This indicates to the user that these buttons are process links that apply to the selected email message.

In Table 1 we summarize all the refactorings we have presented in this section, and classify them by the Intent, Scope and Quality Attributes they address.

5 Case studies and evaluation

We have conducted an evaluation on some websites with the purpose of illustrating the task of finding bad smells and linking them directly to the refactorings in our catalog that bring a cure to the identified bad smells. In the same spirit as Nielsen's book



(a) www.nationwide.co.uk **(b)** www.amazon.com

(c) www.bna.com.ar **(d)** www.lastminute.com

(e) www.tematika.com **(f)** www.amazon.co.uk

Fig. 12 Screenshots of our case studies showing with *orange circles* the bad smells that are described in Table 2. (Color figure online)

on "Homepage Usability" [32], we do not intend to point incompetent site designers but rather to show indicators of the poor state in which web usability still is today, in particular when it comes to how users carry on processes on the Web. We also go a step further and suggest a solution in terms of a specific and cataloged refactoring that corrects each problem. In the second section we evaluate our approach in terms of strengths and limitations.

5.1 Bad smells of six e-commerce Websites and a refactoring proposal

We have analyzed six websites, namely, www.nationwide.co.uk, www.amazon.com, www.bna.com.ar, www.lastminute.com, www.tematika.com, and www.amazon.co.uk.

Table 1 A characterization framework for WBPRs and a summary of our catalogue of WBPRs

Characterization aspects	Web business process refactorings characterization															
	S1	S2	S3	S4	S5	S6	B1	B2	B3	N1	N2	N3	P1	P2	P3	P4
Intent																
Quality in use characteristics (ISO/IEC 25010) [24]																
Effectiveness[a]	X	X			X		X	X		X	X	X	X		X	X
Efficiency[b]	X		X		X	X	X	X	X	X	X			X		X
Satisfaction[c]		X	X	X		X	X	X	X	X	X	X	X	X	X	X
Freedom from Risk[d]		X						X	X	X	X	X		X	X	X
Context coverage[e]		X					X				X					
Scope																
Impacted Web design layers																
Process (structural)	X	X	X	X	X	X		X	X							
Process (behavioral)	X	X		X	X	X	X	X	X							
Navigation	X	X					X	X		X	X	X				
Presentation	X	X					X	X		X	X		X	X	X	X
Modified software artifacts																
Class diagram	X	X	X	X		X		X	X							
Activity diagram / BPMN models	X	X	X	X		X	X	X	X							
Navigation model	X	X					X	X		X	X	X				
Presentation model	X	X					X	X		X	X		X	X	X	X

The analysis we performed had the purpose of finding opportunities for refactoring, i.e., bad smells related to the quality in use of the BP of each website. Note that this analysis may be formalized by measuring quality attributes of interest to final users in the context of a formal framework like that proposed by the Web Quality Evaluation Method (WebQEM) [37]. We have elsewhere proposed an integrated approach that uses WebQEM to find bad smells, then applies refactoring to fix the bad smells, and finally uses WebQEM again to measure the improvement gain [38]. For example, a quality indicator like "Operation grouping cohesiveness" may signal the presence of the bad smell "Poor/Confusing organization of process links" (highlight #8 in Fig. 12), thus indicating the need for a refactoring like "P3) Group process links".

Figure 12 shows a page from each of the considered websites that exhibits some bad smells associated to the execution of activities inside a BP. In the order of the figure, those activities/BPs are: (a) logging in, (b) buying, (c) making a bank transfer, (d) searching for a flight, (e) registering, and (f) checking out. The orange circles on each page point to the bad smells that are listed by number in the second column of Table 2.

Table 2 describes, for each webpage in Fig. 12, the bad smells we found and the refactoring that we can apply to solve each of them. To better illustrate the application of our approach, both the bad smells and the refactorings are not described in general terms as in the previous section but instantiated for the specific example. Moreover, for each refactoring we list a website that shows the solution, as a reference for desirable quality attributes and good practices that users appreciate, and that the refactoring is able to gain, thus improving usability.

5.2 Strengths and limitations of our usability improvement approach

We believe that our approach to improve the quality-in-use properties associated to the access and execution of BPs in e-commerce applications has the following strengths:

- Each refactoring is simple and behavior preserving; therefore applying refactoring is a safe process regarding application functionality.
- It is independent of the method and models adopted to design the BPs; it is also independent of the technologies adopted to realize the application as it works at the conceptual user-centered design level. On this regard, we have purposely presented different refactoring examples using different models for BP representation.
- It provides practical guidance on identifying usability issues and/or opportunities for usability improvements in the business processes as we showed in the examples. This is inherent to the concepts of refactoring, bad smell, and mechanics on which the approach is based.
- Our catalogue of refactorings is extensible. As new bad usability smells are identified and possible refactoring solutions are defined, these can be added to extend the current catalogue. Furthermore, current refactorings can be composed to generate more complex ones.

On the other hand, we are aware of some limitations that we aim to overcome with future work we are pursuing:

Table 2 Summary of bad smells and refactorings in our case studies

Website	Bad smell	Refactoring	Correct solution
(a) www.nationwide.co.uk	(1) A single input field per page; Excessive time to complete login	(S2) Aggregate activities	www.bancofrances.com.ar
(a) www.nationwide.co.uk	(2) Unknown number/description of steps in the login process	(P4) Make explicit the steps (or number) composing a process	www.santander.co.uk (login has 2 numbered steps)
(b) www.amazon.com	(3) Although already logged in, the website requests the user to login again when going to checkout	(S5) Make login data in shopping activity visible to the checkout activity	www.mercadolibre.com.ar
(b) www.amazon.com	(4) No apparent “Sign out”	(N3.a) Improve the description of process links	www.mercadolibre.com.ar
(b) www.amazon.com	(5) No view of current content of shopping cart or total price of items in the cart, unless navigating	(N1) Keep the user up to date of an ongoing process	www.mediashopping.com
(c) www.bna.com.ar	(6) Too many links available without a specific order	(N2) Reduce the number of navigation links provided to the user while executing a process	www.bancofrances.com.ar
(c) www.bna.com.ar	(7) Confusing duplicate links to Print (“Imprimir”) and Download (“Descargar”)	(P2) Remove duplicate process links	www.bancofrances.com.ar
(c) www.bna.com.ar	(8) Confusing organization of links to Add (“Agregar”) and Remove (“Quitar”) in different places of the page	(P3) Group process links that operate on the transfer	www.gmail.com (Fig. 5)
(d) www.lastminute.com	(9) Need to write the whole airport or city name. Need to write it again when making a typo	(B2.d) Add autocomplete in both airport fields	www.lastminute.it (Fig. 4)
(d) www.lastminute.com	(10) Two fields instead of a single selection to complete the dates	(P1) Change day and month widgets by a single calendar widget.	www.lastminute.it (Fig. 4)
(e) www.tematika.com	(11) Premature abandonment of page	(B1.b) Postpone execution of activity	www.saraiva.com.br
(f) www.amazon.co.uk	(12) Inflexibility of the business process to be cancelled.	(B3) Make process cancellable	www.tesco.co.uk

- Our catalogue of refactorings is not exhaustive. As a consequence, our approach to usability improvement might fail to identify a number of usability issues in the analyzed BPs. At the same time, while our refactorings are meant and meaningful for "traditional" e-commerce websites, they might need some adaptation when used in mobile e-commerce applications (M-commerce), as user's usability concerns for the two domains are subtly different [40].
- Our approach has currently limited tool support. At the moment, tool support is provided for the refactoring of the navigation and presentation models of the UWE methodology using the MagicUWE tool [7]. Thus, it requires manual application of refactorings for the process model. However, we are currently developing a tool that will enable to seamlessly introduce any of the refactoring of the subset of navigation and presentation WBPRs by using client-side scripting technologies on the front-end of the application.
- We have not yet performed a fully analytical and quantitative evaluation of the approach. In this regard it is important to state that the improvements in usability become evident from the successful examples that show recurrent and well-known practices in the e-commerce field. However, we have not measured the development effort for realizing each change and we leave this analysis for a further work.

6 Related work

In the context of BPs, several works related to refactoring techniques have been proposed. However, to the best of our knowledge, all of them have a different objective compared to that of our proposal.

The closest work we could find in the literature is that of Zou et al. [54]. The authors claim and verify with an empirical study that the information stored in business process definitions (e.g., task processing sequences and role information) can be leveraged to improve the UI of an existing e-commerce application, and thus its usability. While our work has the same intent (improving usability) and same scope (business processes in e-commerce applications) of that of Zou et al., we propose a completely different approach to reach the intent, i.e. a catalogue of refactorings for the BPs of the application, which can be beneficial for the usability of the application.

The approach described in [27] enables a business architect to establish correspondences between two process models in a systematic way and shows how these correspondences define concrete refactoring operations that serve to improve the "as-is" model. Weber et al. propose a catalog of process model "smells" for identifying refactoring opportunities [52]. In addition, they introduce a set of behavior-preserving techniques for refactoring large process repositories. The refactorings are purely focused on the control-flow perspective in order to improve the internal quality of the process model, but they do not affect the model's semantics or external behavior. In [10], Dijkman et al. propose a technique that can be used to identify four process model refactoring opportunities. The technique is based on metrics that can be used to measure the consistency of activity labels as well as the extent to which processes overlap and the type of overlap that they have. Authors evaluated their technique by applying it to two large process model repositories. Recently, the work of Fernández-Ropero et al.

[12] aims to choose the most appropriate set of business process refactoring operators through the quality assessment concerning understandability and modifiability. These quality features are assessed through well-proven measures proposed in the literature. In these four mentioned approaches, refactoring has been applied in order to improve internal qualities of complex BP models: readability, understandability and maintainability. In this sense, we could say that they propose "BP Refactoring" unlike our work that is focused to improve usability of BPs in e-commerce Web applications. The work of Ferrari et al. promotes a formal approach to refactoring of long running transactions (LRT) [18], [13] represented in signal calculus (SC) so that distributed LRT designed in BPMN can be faithfully represented. On top of SC, the authors define a few refactoring transformations for distributed LRT. Finally, they prove that the given refactoring rules are sound by showing that they preserve (weak) bisimilarity. Workflow graphs are used to model the control flow of BPs in various languages, e.g., BPMN, EPCs and UML (a comparison of BPMN and UML AD is presented in [41]). The approach presented in [51] proposes techniques for automatic workflow graph refactoring and completion. These techniques enable various use cases in modeling and runtime optimization. For example, they allow completing a partial workflow graph, detecting local termination for workflow graphs with multiple ends, and executing models containing OR-joins faster. Some of these techniques are based on workflow graph parsing and Refined Process Structure Tree. This mechanism provides a decomposition of a workflow graph into a hierarchy of sub-workflows that are subgraphs with a single entry and a single exit of control. Such a decomposition is the crucial step, for example, to translate a process modeled in a graph-based language such as BPMN into a process modeled in a block-based language such as BPEL. It is desirable that the decomposition be *unique*, *modular* and fine, where modular means that a local change of the workflow graph can only cause a local change of the decomposition [50].

On the other hand, without specifically addressing the refactoring concept, a number of approaches deal with the critical issue of improving BP quality. Among others, Grigori et al. [21] pursue this goal through an approach that analyzes, predicts and prevents the occurrences of exceptions in the BP, i.e., derivations from the desired or acceptable behavior. They characterize the problem and propose a solution, based on data warehousing and mining techniques. Tilley et al. present in [48] a process to reengineer Web application transactions (i.e., BPs implemented by a Web application) that consists of recovering the "as-is" design model of the transaction, analyzing it to determine opportunities for restructuring, and redesign the transaction accordingly. Similar to our work, the goal of the proposed reengineering process is to emerge with a transaction design that better reflects the user experience and also facilitates disciplined evolution of the Web-based application. The authors, however, do not propose any catalogue of possible restructuring actions as they focus on the description of the steps composing the reengineering process. In [53], a set of 18 change patterns and seven change support features are suggested in order to enhance flexibility in Process-Aware Information Systems (PAISs). Based on the proposed change patterns and features, authors provide a detailed analysis and evaluation of selected approaches from both academia and industry. The work facilitates the selection of technologies for realizing flexible PAISs. Finally, the PAISs usually support BP design by means of graphical graph-oriented BPMLs in conjunction with textual executable specifications. In [9] authors discuss the flexibility of different BPMLs which are the main interface for users that need to change the behavior of PAISs. In particular, they show how common BPMLs features, that seem good when considered alone, have a negative impact on flexibility when they are combined together for providing a complete executable specification. A model has to be understood before being changed and a change is made only when the benefits outweigh the effort. Two main factors have a great impact on comprehensibility and ease of change: concurrency and modularity. They show why BPMLs usually offer a limited concurrency model and lack of modularity and, finally, they discuss how to overcome these problems.

Unlike the works mentioned above, our approach is focused on improving *quality-in-use* characteristics-such as *satisfaction, efficiency* and *efficacy*- and, overall, *usability*, related to the execution of BPs in e-commerce Web applications. To reach this goal we propose and describe the application of a set of refactorings for the different layers in which a BP is designed in a business Web application.

7 Conclusions and future work

Since its introduction in the 1990's, refactoring has demonstrated to be a powerful technique for dealing with software decay and to improve internal and external quality attributes of a software system. Originally conceived to make a codebase easy to understand and maintain, during the years, refactoring has in fact been extended in intent and scope and today it is applied with success to refactor a variety of software artifacts at various levels of abstraction and with different intents.

After having introduced refactoring in the realm of Web application design models [15–17], in this article we concentrated on BPs in e-commerce applications, and proposed a definition and a catalogue of refactorings that can be applied to improve quality-in-use characteristics, such as effectiveness, efficacy, user satisfaction, and, overall, usability, related to the execution of BPs in Web applications.

To make presentation concrete and provide evidence of the benefits that applying our WBPRs can bring, we provided examples for each of them with reference to well-known e-commerce websites. To make the presentation systematic, we presented and proposed a framework for characterizing WBPRs based on their intent and scope.

For a representative subset of WBPRs, we illustrated in detail how the need and/or opportunity for refactoring can be identified (bad smell) and how the refactorings can be applied (mechanics), by showing the "as-is" and "to-be" versions of the associated software artifacts using different design notations for BP design. The whole refactoring approach and each WBPR, however, are applicable no matter what the notation and the design method used to model the BP are.

Though conceived for Web applications, we believe that our WBPRs, particularly those classified as structural and behavioral WBPRs, can be profitably applied also to traditional, i.e. not Web-based, business applications, as they represent changes for the process organization and control-flow, independently of the process implementation. Moreover, they can be applied to the more general field of process workflows involving not only software, though in this paper we have concentrated in the software part of these systems.

Future work we aim to develop will be devoted to extending our catalogue of WBPRs, which now includes eighteen refactorings, and to developing tools and extended guidelines, including a checklist, to support and partially automate the whole refactoring process. We are also analyzing those refactorings that involve transformations using the kind of interactions popular in rich internet applications (RIAs) as in the example of Fig. 1, since even some of the refactorings in our catalogue may have different realizations in the running application, such as those commonly found in RIA. Our guidelines and checklist will also support these implementation "variants" and suggestions on when to use them (e.g., according to the application nature). Finally, we are researching on how our refactorings affect automatic interaction tests by enriching our approach for test-driven development of Web applications [43] with the catalogue of refactorings, which are expressed as transformations in the interface and in the interaction requirements of the application.

References

1. Aberer, K., Datta, A., Despotovic, Z., & Wombacher, A. (2003). Separating business process from user interaction in web-based information commerce. *Electronic Commerce Research*, *3*(1), 83–111.
2. AGConsult (2009). 48% of visitors on e-commerce websites don't buy due to lack of usability. http://webusability-blog.com/48-of-visitors-on-e-commerce-websites-dont-buy-due-to-lack-of-usability/.
3. Ambler, S. (2012). User interface flow diagrams (Storyboards). http://www.agilemodeling.com/artifacts/uiFlowDiagram.htm.
4. Bevan, N. (1999). Quality in use: Meeting user needs for quality. *Journal of Systems and Software*, *49*(1), 89–96.
5. Boger, M., Sturm, T., & Fragemann, P. (2003). *Refactoring browser for UML. Objects, components, architectures, services, and applications for a networked world* (pp. 366–377)., Lecture notes in computer science Berlin: Springer.
6. Brambilla, M., Ceri, S., Fraternali, P., & Manolescu, I. (2006). *Process modeling in web applications. ACM transactions on software engineering and methodology (TOSEM)*. New York: ACM Press.
7. Busch, M., & Koch, N. (2009). MagicUWE—A CASE tool plugin for modeling web applications. In: *Proceedings of the 9th international conference on web engineering (ICWE 2009)* (pp. 505–508). Berlin: Springer.
8. Casalo, L., Flavian, C., & Guinaliu, M. (2008). The role of usability and satisfaction in the consumer's commitment to a financial services website. *International Journal of Electronic Finance*, *2*(1), 31–49.
9. Combi, C., & Gambini, M. (2009). Flaws in the flow: The weakness of unstructured business process modeling languages dealing with data. *Proceeding of the confederated international conferences, CoopIS, DOA, IS, and ODBASE 2009 (OTM '09) on the move to meaningful internet systems: Part I*. Lecture notes in computer science, 5870 LNCS (Part 1) (pp. 42–59). Berlin: Springer.
10. Dijkman, R., Gfeller, B., Küster, J., & Völzer, H. (2011). Identifying refactoring opportunities in process model repositories. *Information and Software Technology*, *53*, 937–948.
11. Distante, D., Rossi, G., Canfora, G., & Tilley, S. (2007). A comprehensive design model for integrating business processes in web applications. *International Journal of web Engineering and Technology (IJWET)*, *3*(1), 43–72.
12. Fernández-Ropero, M., Pérez-Castillo, R., Caballero, I., & Piattini, M. (2012). Quality-driven business process refactoring. *World Academy of Science, Engineering and Technology, Issue*, *66*, 960–966.
13. Ferrari, G. L., Guanciale, R., Strollo, D., & Tuosto, E. (2009). Refactoring long running transactions. In: *Proceedings of the 6th international workshop on web services and formal methods (WS-FM 2008)* (Vol. 5387, pp. 207–223)., Lecture notes in computer science Heidelberg: Springer.
14. Fowler, M. (1999). *Refactoring: Improving the design of existing code*. Boston: Addison-Wesley.
15. Garrido, A., Rossi, G., & Distante, D. (2007). Model refactoring in web applications. In: *Proceedings of the 9th IEEE international symposium on web site evolution (WSE 2007)*. New York: IEEE Computer Society.

16. Garrido, A., Rossi, G., & Distante, D. (2009). Systematic improvement of web applications design. *Journal of Web Engineering*, *8*(4), 371–404.
17. Garrido, A., Rossi, G., & Distante, D. (2011). *Refactoring for Usability in Web Applications*. IEEE Software, May/June 2011 (pp. 31–38). New York: IEEE Computer Society.
18. Grayand, J., & Reuter, A. (1993). *Transaction processing: Concepts and techniques*. San Francisco: Morgan Kaufmann.
19. Green, D. T., & Pearson, J. M. (2011). Integrating website usability with the electronic commerce acceptance model. *Behaviour & Information Technology*, *30*(2), 181–199.
20. Gregg, D. G., & Walczak, S. (2010). The relationship between website quality, trust and price premiums at online auctions. *Electronic Commerce Research*, *10*(1), 1–25.
21. Grigori, D., Casati, F., Dayal, U., & Shan, M.-C. (2001). Improving business process quality through exception understanding, prediction, and prevention. In *Proceedings of the 27th Very Large Data Base Conference (VLDB '01)*. San Francisco, CA: Morgan Kaufmann Publishers Inc.
22. Harold, E. R. (2008). *Refactoring HTML: Improving the design of existing web applications*. Boston: Addison-Wesley.
23. ISO 9241–11:1998 (1998). *Ergonomic requirements for office work with visual display terminals (VDTs)—Part 11: Guidance on usability*. Geneva: ISO Copyright Office.
24. ISO/IEC 25010:2011 (2011). Systems and software engineering—Systems and software Quality requirements and evaluation (SQuaRE)—System and software quality models. Geneva: ISO Copyright Office.
25. Kateb, D. E., Mouelhi, T., Le Traon, Y., Hwang, J. H., & Xie, T. (2012). Refactoring access control policies for performance improvement. In: *Proceedings of the 3rd ACM/SPEC international conference on performance Engineering (ICPE'12)* (pp. 323–334). New York: ACM Press.
26. Koch, N., Kraus, A., Cachero, C., & Meliá, S. (2004). Integration of business processes in web applications. *Journal of Web Engineering*, *3*(1), 22–49.
27. Küster, J. M., Koehler J., Ryndina, K. (2006). Improving Business Process Models with Reference Models in Business-Driven Development. In: *Proceedings of the international business process management workshops 2006*. Lecture notes in computer science (Vol. 4103/2006, pp. 35–44). Berlin: Springer.
28. Lauesen, S. (2005). *User interface design—a software engineering perspective—the Virtual Windows method*. Boston: Addison-Wesley.
29. Liu, C.-H., Kung, D. C., Hsia, P., & Hsu, C.-T. (2000). Object-based data flow testing of web applications. In: *Proceedings of the First Asia-Pacific Conference on Quality Software (APAQS '00)*. Washington, DC: IEEE Computer Society.
30. Nathan, R. J., & Yeow, P. H. P. (2010). Crucial web usability factors of 36 industries for students: a large-scale empirical study. *Electronic Commerce Research*, *11*(2), 151–180.
31. Nielsen, J. (1999). *Designing web usability*. New York: New Riders Publishing.
32. Nielsen, J. (2001). *Homepage usability: 50 websites deconstructed*. New York: New Riders Publishing.
33. Nielsen, J. (2011). E-commerce usability. http://www.useit.com/alertbox/ecommerce.html.
34. Nielsen, J. (2001). Did poor usability kill E-Commerce?. http://www.useit.com/alertbox/20010819.html.
35. Object Management Group (2010). Unified modeling language version 2.3. http://www.omg.org/spec/UML/2.3/.
36. Object Management Group (2011). Business process modeling and notation (BPMN) version 2.0. http://www.omg.org/spec/BPMN/2.0/PDF
37. Olsina, L., & Rossi, G. (2002). Measuring web application quality with WebQEM. *IEEE Multimedia*, *9*, 20–29.
38. Olsina, L., Garrido, A., Distante, D., & Canfora, G. (2008). Web application evaluation and refactoring: A quality-oriented improvement approach. *Journal of Web Engineering*, *7*(4), 258–280.
39. Opdyke, W., & Johnson, R. (1993). Creating abstract superclasses by refactoring. In: *Proceedings of the 1993 ACM conference on computer science (CSC 93)* (pp. 66–73). New York: ACM Press.
40. Ozok, A., & Wei, J. (2010). An empirical comparison of consumer usability preferences in online shopping using stationary and mobile devices: results from a college student population. *Electronic Commerce Research*, *10*(2), 111–137.
41. Peixoto, D. C. C., Batista, V. A., Atayde, A. P., Borges, E. P., Resende, R. F., Isaías, C., et al. (2008). A Comparison of BPMN and UML 2.0 activity diagrams. VII Simpósio Brasileiro de Qualidade de

Software (SBQS 2008). Florianópolis, SC: Sociedade Brasileira de Computação – SBC, ISBN:978-85-7669-180-8.
42. Pourshahid, A., Amyot, D., Peyton, L., Ghanavati, S., Chen, P., Weiss, M., et al. (2009). Business process management with the user requirements notation. *Electronic Commerce Research*, *9*(4), 269–316.
43. Robles Luna, E., Grigera, J., & Rossi, G. (2009). Bridging test and model-driven approaches in web engineering. In: *Proceeding of the 9th international conference in web engineering (ICWE 2009)*. Lecture notes in computer science 5648 Springer 2009 (pp. 136–150). Heidelberg: Springer.
44. Schäfer, M., de Moor, O. (2010). Specifying and Implementing Refactorings. In: *Proceedings of the 25th ACM international conference on object oriented programming systems languages and applications (OOPSLA '10)* (pp. 286–301). New York: ACM Press.
45. Schaffer, E., & Sorflaten, J. (1998). Web usability illustrated: Breathing easier with your useable E-commerce site. *EDI Forum: The Journal of Electronic Commerce*, 11(4), pp. 50–52, 57–64.
46. Schmid, H., & Rossi, G. (2004). Modeling and designing processes in E-commerce applications. *IEEE Internet Computing*, *8*(1), 19–27.
47. Shneiderman, B., Plaisant, C., Cohen, M., & Jacobs, S. (2009). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (5th edition). Englewood Cliffs: Prentice Hall.
48. Tilley, S., Distante, D., & Huang, S. (2004). Web site evolution via transaction reengineering. In: *Proceedings of the 6th international workshop on web site evolution (WSE 2004)*. Los Alamitos, CA: IEEE Computer Society Press.
49. Torres, V., & Pelechano, V. (2006). Building business process driven web applications. In:*Proceedings of business process management 2006* (pp. 322–337). Berlin: Springer.
50. Vanhatalo, J., Völzer, H., & Koehler, J. (2008). The Refined process structure tree. In:*Proceedings of the 6th international conference on business process management (BPM '08)* (Vol. 5240, pp. 100–115). Lecture notes in computer science. Heidelberg: Springer.
51. Vanhatalo, J., Völzer, H., Leymann, F., & Moser, S. (2008). Automatic workflow graph refactoring and completion. In: *Proceedings of the 6th international conference on service-oriented computing (ICSOC '08)* (pp. 100–115). Lecture notes in computer science, 5364 LNCS. Heidelberg: Springer.
52. Weber, B., Reichert, M., Mendling, J., & Reijers, H. A. (2011). Refactoring large process model repositories. *Computers in Industry*, *62*(5), 467–486.
53. Weber, B., Reichert, M., & Rinderle-Ma, S. (2008). Change patterns and change support features—enhancing flexibility in process-aware information systems. *Data & Knowledge Engineering*, *66*, 438–466.
54. Zou, Y., Zhang, Q., & Zhao, X. (2007). Improving the usability of E-commerce applications using business processes. *IEEE Transactions Software Engineering*, *33*, 837–855.

Damiano Distante is Assistant Professor of Computer Science and Engineering at the Unitelma Sapienza University of Rome, Italy. His main field of research is software engineering in general and Web engineering and software evolution in particular. His research interests include: model-driven development of Web applications, Web systems evolution, and the application of text mining and information retrieval techniques to various domains, such as elearning. He is author of more than forty publications in international journals and proceedings of international conferences. He has a PhD in Information Engineering from the University of Salento, Italy. He is member of the IEEE Computer Society.

Alejandra Garrido is an Assistant Professor at the Facultad de Informática, Universidad Nacional de La Plata, Argentina, where she's a member of the Research and Development in Advanced IT Lab (LIFIA). She is also a researcher at Argentina's National Scientific and Technical Research Council (CONICET). Her research interests include refactoring and Web engineering, focusing on design patterns, frameworks, refactoring for the C language, and refactoring for usability. Garrido has a PhD in computer science from the University of Illinois at Urbana-Champaign. She's a member of the Hillside Group.

Julia Camelier-Carvajal is a Master student of the Facultad de Informática, Universidad Nacional de La Plata, Argentina. Her research is mainly focused on refactoring and Web engineering. She is currently working in the private industry as a project manager.

Roxana Giandini is Assistant Professor at the Facultad de Informática, Universidad Nacional de La Plata, Argentina. She is a member of the Research and Development in Advanced IT Lab (LIFIA). Her research area is software modeling and software engineering. Those areas are closely related to formal methods, that's why she also interested in these topics. Giandini has a PhD in Computer Science from the Universidad Nacional de La Plata, in the area of Model-Driven Software Development.

Gustavo Rossi is Full Professor at the Facultad de Informàtica, Universidad Nacional de La Plata, Argentina, head of the Research and Development in Advanced IT Lab (LIFIA), and a researcher of the Argentina's National Scientific and Technical Research Council (CONICET). His research interests include agile approaches for Web applications development and model-driven development. Rossi is one of the developers of the mature OOHDM Web design approach and has authored more than a hundred publications in the field of Web and e-commerce applications design. He has a PhD in computer science from the Pontifícia Universidade Católica do Rio de Janeiro, Brazil and he is a member of ACM and IEEE.