The Journal of Systems and Software 105 (2015) 107–124

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Design and programming patterns for implementing usability functionalities in web applications

Francy D. Rodríguez [a,*], Silvia T. Acuña [b], Natalia Juristo [a]

[a] Escuela Técnica Superior de Ingenieros Informáticos, Universidad Politécnica de Madrid, Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid, Spain
[b] Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Calle Francisco Tomás y Valiente 11, 28049 Madrid, Spain

ARTICLE INFO

Article history:
Received 31 July 2014
Revised 31 March 2015
Accepted 2 April 2015
Available online 11 April 2015

Keywords:
Software engineering
Design patterns
Programming patterns

ABSTRACT

Usability is a software system quality attribute. There are usability issues that have an impact not only on the user interface but also on the core functionality of applications. In this paper, three web applications were developed to discover patterns for implementing two usability functionalities with an impact on core functionality: Abort Operation and Progress Feedback. We applied an inductive process in order to identify reusable elements to implement the selected functionalities. For communication purposes, these elements are specified as design and programming patterns (PHP, VB.NET and Java). Another two web applications were developed in order to evaluate the patterns. The evaluation explores several issues such as ease of pattern understanding and ease of pattern use, as well as the final result of the applications.

We found that it is feasible to reuse the identified solutions specified as patterns. The results also show that usability functionalities have features, like the level of coupling with the application or the complexity of each component of the solution, that simplify or complicate their implementation. In this case, the Abort Operation functionality turned out to be more feasible to implement than the Progress Feedback functionality.

1. Introduction

Usability is a critical software system quality attribute in highly interactive systems (Juristo et al., 2007a). Usability is defined in ISO Standard 9241-210 (ISO, 2010) as "the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". SE originally considered that a satisfactory level of usability could be achieved by including usability features in the design of the user interface (UI). In this scheme of things, it was sufficient to use strategies that separated the UI from the core functionality of the applications.

It was later established that the separation strategy is not good enough to output a usable system, and there are usability issues that should be tackled as of the early development process activities (Juristo et al., 2007a) because they affect the core functionality of the applications. A usability issue with impact on the software system core that is not taken into account early on in the development process will generate high costs, and the new system is unlikely to implement all its features (John et al., 2009).

The literature abounds with studies that deal with usability in early development process activities and present high-level solutions. Some of the proposals for including usability in software development are introduced as guidelines or patterns. For example, Juristo et al. (2007b) propose guidelines for eliciting requirements and Bass et al. (2001) and Folmer et al. (2003) introduce architectural patterns for including usability functionalities such as aggregating commands, cancelling commands, predicting task duration and verifying resources. Other approaches tackle activities later on in the development process. Thus, Juristo et al. (2007a) analyze the impact of usability issues on detailed design, and Folmer et al. (2006) present final implementations as an example to help establish the implications of usability for system architecture.

This research is a continuation of the effort to address usability issues that affect software system functionality within the SE development process. The difference is that it targets the later activities of the development process not normally addressed in the literature. We set out to establish whether it is possible to find reusable detailed design and programming solutions in order to build applications that implement usability functionalities. In this study we also analyze whether the identified reusable solutions can be specified as design patterns and programming patterns (D&P patterns). Finally, independent developers use the proposed solutions to implement systems for the purposes of evaluation.

* Corresponding author. Tel.: +34693367642.
E-mail addresses: fd.rodriguez@alumnos.upm.es, netfrancy@hotmail.com (F.D. Rodríguez), silvia.acunna@uam.es (S.T. Acuña), natalia@fi.upm.es (N. Juristo).

http://dx.doi.org/10.1016/j.jss.2015.04.023
0164-1212/

We selected two usability functionalities called usability mechanisms (UM), which have a major impact on design (Juristo et al., 2007a): Abort Operation (AO) and Progress Feedback (PF). These two functionalities cannot be implemented focusing on the UI only. The study is limited to web applications. Web applications differ from other application types in that the client side is composed of dynamic web pages which are interpreted by a browser and generate particular reuse conditions. A web page can be created on the server side or client side depending on the programming type or technologies used (W3C, 2014).

The object-oriented design and programming paradigm and three different server-side languages are used: PHP, VB.NET and Java. Object-oriented programming encourages reuse (Szyperski, 2002), and, as all three development projects use the same type of elements, we can look for the elements that they have in common. Although PHP was not originally an object-oriented language, its latest versions provide for the design and use of classes and methods. The web client side uses the Javascript language.

This paper was structured as follows. Section 2 presents the background and work related to our proposal. Section 3 describes the research method applied in order to both identify and evaluate the reusable elements. Section 4 shows the reusable elements discovered by the research and their specification as patterns for the AO and PF UMs. Section 5 describes how the proposed patterns were evaluated. Section 6 discusses the results and their evaluation. Section 7 presents the conclusions and future work.

2. Background

2.1. Usability mechanisms

The field of human–computer interaction (HCI) has addressed system usability at length. HCI guidelines are useful for achieving a satisfactory level of system usability. HCI researchers have defined a great many patterns bearing different names: interaction or interaction design patterns (Tidwell, 2010; Welie and Trætteberg, 2000), user interface patterns (Laakso, 2003), usability patterns (Brighton, 1999; Perzel and Kane, 1999), and web design patterns (Van Duyne et al., 2006). All these patterns have in common that they offer solutions to specific usability problems, although they are described or grouped differently. There are also several pattern libraries for user interface design built by companies and available on the web (Yahoo, 2013; Pattern Factory Oy, 2014; Infragistics, 2015; Toxboe, 2015).

Based on HCI recommendations about how to improve software systems usability, Juristo et al. (2007a) identified three categories of recommendations depending on their effect on software development: usability recommendations with an impact on the UI, usability recommendations with an impact on the development process and usability recommendations with an impact on design. They reported empirical evidence of the relationship between usability and software design, identified functional usability features (FUF) with a high impact on design and measured their impact on real-world applications. The identified functionalities are a product of the HCI recommendations. In turn, each HCI author identifies different FUF subtypes. Each subtype has been referred to as UM and has a name indicating its functionality. A non-exhaustive list of FUFs and their respective mechanisms is presented in Juristo et al. (2007b). Table 1 shows the identified usability features and their respective mechanisms.

The use of the term usability functionality is potentially controversial, as usability is typically construed as being a non-functional requirement. However, Juristo et al. (2007b) established that the features listed in Table 1 "represent particular functionalities that can be built into a software system to increase usability. Since functional requirements describe the functions that the software is to execute, we consider that the usability features in Table 1 should be treated as functional requirements (even though they are usability-related requirements). Such functional usability requirements need to be explicitly specified, just like any other functionality". Previous research by Bosch and Juristo (2003) and Bass et al. (2004) had already demonstrated the relationship between usability and software system functionalities.

In this paper, we propose D&P patterns to implement two of the UMs listed in Table 1. There are another two papers based on the usability functionalities and mechanisms described in Table 1. The aim of both papers is to add usability functionalities to software systems, but they take completely different approaches. One of the approaches (Carvajal et al., 2013) proposes guidelines for developers to incorporate FUFs into each development process activity from the requirements elicitation to the design stages. The second approach (Panach et al., 2014) is an extension of Juristo et al.'s research for model-driven development (MDD). Their aim is to build usability functionalities into software products developed using MDD.

We selected two UMs: Abort Operation, part of the Undo/Cancel FUF, and Progress Feedback, part of the Feedback FUF. Both UMs are highlighted in gray in Table 1. The other mechanisms belonging to these two FUFs are Global Undo, Object-Specific Undo and Go Back

Table 1
Usability mechanisms with an impact on software design.

Usability feature	Usability mechanism	Goal
Feedback	System status	To inform users about the internal status of systems.
	Interaction	To inform users that the systems has registered a user interaction, i.e. that the system has heard the user.
	Warning	To inform users of any action with important consequences.
	Progress feedback	To inform users that the system is processing an action that will take some time to complete.
Undo/Cancel	Global undo	To undo system actions at several levels.
	Object-specific undo	To undo several actions on an object.
	Abort operation	To cancel the execution of an action or the whole application.
	Go back	To go back to a particular state in a command execution sequence.
User input error prevention/correction	Structured text entry	To help prevent the user from making data input errors.
Wizard	Step-by-step execution	To help users to do task that require different steps with user input and correct such input.
User profile	Preferences	To record each user's options for using system functions.
	Personal object space	To record each user's options for using the system interface.
	Favourites	To record certain places of interest for the user.
Help	Multilevel help	To provide different help levels for different users.
Command aggregation	Command aggregation	To express possible actions to be taken with the software through commands that can be built from smaller parts.