Information and Software Technology 54 (2012) 331–346

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

A methodology to assess the impact of design patterns on software quality

Apostolos Ampatzoglou [a,*], Georgia Frantzeskou [b], Ioannis Stamelos [a]

[a] Department of Informatics, Aristotle University, Thessaloniki, Greece
[b] Information and Communication Systems Engineering Department, University of the Aegean, Samos, Greece

ARTICLE INFO

Article history:
Received 8 June 2011
Received in revised form 24 October 2011
Accepted 29 October 2011
Available online 4 November 2011

Keywords:
Structural quality
Design patterns
Object-oriented metrics
Quality

ABSTRACT

Context: Software quality is considered to be one of the most important concerns of software production teams. Additionally, design patterns are documented solutions to common design problems that are expected to enhance software quality. Until now, the results on the effect of design patterns on software quality are controversial.
Aims: This study aims to propose a methodology for comparing design patterns to alternative designs with an analytical method. Additionally, the study illustrates the methodology by comparing three design patterns with two alternative solutions, with respect to several quality attributes.
Method: The paper introduces a theoretical/analytical methodology to compare sets of "canonical" solutions to design problems. The study is theoretical in the sense that the solutions are disconnected from real systems, even though they stem from concrete problems. The study is analytical in the sense that the solutions are compared based on their possible numbers of classes and on equations representing the values of the various structural quality attributes in function of these numbers of classes. The exploratory designs have been produced by studying the literature, by investigating open-source projects and by using design patterns. In addition to that, we have created a tool that helps practitioners in choosing the optimal design solution, according to their special needs.
Results: The results of our research suggest that the decision of applying a design pattern is usually a trade-off, because patterns are not universally good or bad. Patterns typically improve certain aspects of software quality, while they might weaken some other.
Conclusions: Concluding the proposed methodology is applicable for comparing patterns and alternative designs, and highlights existing threshold that when surpassed the design pattern is getting more or less beneficial than the alternative design. More specifically, the identification of such thresholds can become very useful for decision making during system design and refactoring.

1. Introduction

Object oriented design patterns have been introduced in mid 1990s as a catalog of common solutions to common design problems, and are considered as standard of "good" software designs [31]. The notion of patterns was firstly introduced by Alexander et al. [2] in the field of architecture. Later the notion of patterns has been transformed in order to fit software design by Gamma, Helm, Johnson and Vlissides (GoF) [31]. The authors cataloged 23 design patterns, classified according to two criteria. The first, i.e. purpose, represents the motivation of the pattern. Under this scope patterns are divided into creational, structural and behavioral patterns. The second criterion, i.e. scope, defines whether the pattern is applied on object or class level. In [31], the authors suggest that using specific software design solutions, i.e. design patterns, provide easier maintainability and reusability, more understandable implementation and more flexible design. At this point it is necessary to clarify GoF are not the first or the only design patterns in software literature. Some other well known patterns are architectural patterns, computational patterns, game design patterns, etc. In recent years, many researchers have attempted to evaluate the effect of GoF design patterns on software quality. Reviewing the literature on the effects of design pattern application on software quality provides controversial results. Until now, researchers attempted to investigate the outcome of design patterns with respect to software quality through empirical methods, i.e. case studies, surveys and experiments, but safe conclusions cannot be drawn since the results lead to different directions. As mentioned in [37,39,53,59,69], design patterns propose elaborate design solutions to common design problems that can be implemented with simpler solutions as well.

In this paper we propose a methodology for comparing pattern designs. The proposed method is analytical in the sense that

* Corresponding author.
E-mail addresses: apamp@csd.auth.gr (A. Ampatzoglou), gfran@aegean.gr (G. Frantzeskou), stamelos@csd.auth.gr (I. Stamelos).

0950-5849/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2011.10.006

system comparison is not performed on specific system instances, but on design motifs that can describe every possible instance of a system during its extension. The methodology is generic so that it can be applied for comparing design alternatives that describe equivalent functionality and have specified axes of change. The proposed method attempts to formulate several quality attributes as functions of functionality addition on multiple, equivalent solutions to a design problem. Then the functions are compared so as to identify cut-off points during system maintenance when one solution gets better or worse than the other. In this study we illustrate the applicability of the methodology by employing it for comparing GoF design patterns to equivalent adhoc solutions. In the next section, we present background information on design patterns. In Section 3, the methodology is presented and in Section 4, three exploratory cases are demonstrated. A discussion on the methodology is presented in Section 5. Conclusions, threats to validity and future work are presented by the end of the paper.

2. Related work

In this section of the paper, previous scientific research related to design patterns is presented. More specifically, the section is organized into paragraphs concerning indications on pattern identification according to metric scores, qualitative evaluation of pattern employment, quantitative evaluation of pattern employment, discussion on pattern application and class change proneness, results of controlled experiments on the comparison of design pattern application versus simpler solutions with respect to code maintainability, research on pattern formalization, systematic pattern selection and finally the use of metrics for measuring design structures.

2.1. Metrics as indicators of pattern existence

Firstly, in several papers [5,32,33,53] it is implied that object oriented software metrics can provide indications on the necessity or emergence of specific design patterns. In [5], the authors compute several metrics (number of public, private and protected attributes, number of public, private and protected operations, number of associations, aggregations and inheritance, total number of attributes, methods and relations) so as to get indications on the existence of five structural patterns (Adapter, Proxy, Composite, Bridge and Decorator). Empirical results on the methodology show that software metrics are essential in order to reduce the problem search space and therefore enhance the proposed design pattern detection algorithm. In [53], the authors attempt to introduce some metrics for conditional statements and inheritance trees so as to provide indications for the necessity of applying design patterns, in a low quality design, through refactoring. The proposed methodology, apart from identifying the need for a pattern in a specific set of classes, provides suggestions concerning the pattern that should be applied for solving the problem. Gueheneuc et al. [32], propose a methodology on identifying design motif roles through the use of object-oriented metrics. The suggested fingerprints are ranges of metric scores that imply the existence of design motif role. The authors have chosen to use size, cohesion and coupling metrics, while the patterns under consideration are Abstract Factory, Adapter, Builder, Command, Composite, Decorator, Factory Method, Iterator, Observer, Prototype, Singleton, State, Strategy, Template Method and Visitor. In [33] the authors improve their identification process by combining a structural and a numerical approach. This fact leads to the identification of both complete and incomplete pattern instances, and the reduction of false positive results.

2.2. Qualitative evaluation of design patterns

Additionally, several studies attempted to qualitatively evaluate the effects of object-oriented design patterns on software quality. According to McNatt and Bieman [55], the authors claim that patterns should be considered as structural units and therefore issues such as cohesion and coupling between the patterns should be examined. More specifically, the couplings between patterns are characterized as “loose” and as “tight” and their benefits and costs with respect to maintainability, factorability and reusability are being examined. Although the paper introduces several coupling types, namely intersection, composite and embedded, the way that they are demarcated is not clear. Specifically, there is a default coupling category, i.e. intersection, and any type of coupling that does not fit any other group is classified in the default category. In [75], the author presents an industrial case study, where inappropriate pattern application has led to severe maintainability problems. The reasons of inappropriately using design patterns have been classified into two categories, i.e. (1) software developers have not understood the rationale behind the patterns that they have employed and (2) the patterns that have been employed have not met project requirements. Additionally, the paper highlights the need for documenting pattern usage and the fact that pattern removal is extremely costly. In [42], the authors investigate the correlation among the quality of the class and whether the class play any roles in a design pattern instance. The results suggest that there are several differences in quality characteristics of classes that participate in patterns.

2.3. Quantitative evaluation of design patterns

Furthermore, regarding quantitative evaluation of design pattern application, in [39], the author attempts to demonstrate the effect of three design patterns (Mediator, Bridge and Visitor) on metric scores of three different categories (coupling, inheritance and size). According to the paper, there are several metric thresholds that, when surpassed, the pattern application is beneficial. The study’s methodology is solid since it is based on pattern definitions and mathematical equations, but it deals with only one metric per pattern. Additionally, in [38], the authors have investigated the effect of the patterns on one quality attribute, i.e. the most obvious quality attribute that the pattern has effect on. The selection of the quality attribute has been based on the pattern’s non functional requirements, whereas the selection of the metric has been based on [8]. The drawback of this research is that it does not take into account possible trade-offs that pattern usage induces. For example, when a pattern is employed, the coupling of the system may decrease, but as a side effect the size may increase. If a quality attribute is related to size and coupling, drawing a conclusion that this attribute is enhanced because of the decrease in coupling is not safe.

In [4], the authors attempt to investigate the effect of design pattern application in game development through a case study. The results of the case study are both qualitative and quantitative, but the domain of the research is limited to games and therefore results cannot be generalized. The patterns under study are Bridge and Strategy, whereas the considered metric categories are size, complexity, coupling and cohesion. The results of the case study suggest that pattern application enhance cohesion, coupling and complexity metrics, but as a side effect the size metrics increase. In [41], Khomh et al. performed a survey with professional software engineers. The results of the empirical study suggest that design patterns do not always impact software quality positively. More specifically, the negatively influenced quality issues are suggested to be simplicity, learnability and understandability. However, as it is referenced in the paper, marginal results (e.g.