Information and Software Technology 66 (2015) 58–72

Contents lists available at ScienceDirect

Information and Software Technology

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

Quantitative analysis of fault density in design patterns: An empirical study

Mahmoud O. Elish *, Mawal A. Mohammed

Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

ARTICLE INFO

Article history:
Received 31 October 2014
Received in revised form 24 May 2015
Accepted 26 May 2015
Available online 1 June 2015

Keywords:
Design patterns
Software quality
Fault density
Quantitative analysis

ABSTRACT

Context: There are many claimed advantages for the use of design patterns and their impact on software quality. However, there is no enough empirical evidence that supports these claimed benefits and some studies have found contrary results.
Objective: This empirical study aims to quantitatively measure and compare the fault density of motifs of design patterns in object-oriented systems at different levels: design level, category level, motif level, and role level.
Method: An empirical study was conducted that involved five open-source software systems. Data were analyzed using appropriate statistical test of significance differences.
Results: There is no consistent difference in fault density between classes that participate in design motifs and non-participant classes. However, classes that participate in structural design motifs tend to be less fault-dense. For creational design motifs, it was found that there is no clear tendency for the difference in fault density. For behavioral design motifs, it was found that there is no significant difference between participant classes and non-participant classes. We observed associations between five design motifs (Builder, Factory Method, Adapter, Composite and Decorator) and fault density. At the role level, we found that only one pair of roles (Adapter vs. Client) shows a significant difference in fault density.
Conclusion: There is no clear tendency for the difference in fault density between participant and non-participant classes in design motifs. However, structural design motifs have a negative association with fault density. The Builder design motif has a positive association with fault density whilst the Factory Method, Adapter, Composite, and Decorator design motifs have negative associations with fault density. Classes that participate in the Adapter role are less dense in faults than classes that participate in the Client role.

1. Introduction

Design Patterns (DPs) are generic solutions to common design problems. The objective of cataloging these solutions, including their intent, motivation, applicability, structure, participants, consequences, etc. is to make them reusable. Gamma et al. [14] classified DPs (known as GoF DPs) into three categories: creational patterns, structural patterns, and behavioral patterns. Creational patterns are concerned with creating collections of objects in flexible ways. Structural patterns are concerned with representing collections of related objects. Behavioral patterns are concerned with capturing behavior among collections of objects. There are 23 GoF DPs: five creational patterns, seven structural patterns, and 11 behavioral patterns. Design motifs refer to the solution parts of DPs that are disseminated in the source code of the systems in which DPs are applied [28,39]. In a design motif, there is one or more participant classes that play different roles.

Since their introduction, DPs have attracted the attention of software researchers and practitioners due to the claimed advantages of their application. They are claimed to improve programmers' productivity, promote best design practices, help novice designers to acquire more experience in software design, and make communication easier among team members. Despite these claims, the impact of DPs on software quality is still a debatable issue. Zhang and Budgen [49] conducted a survey of experienced users' perceptions about DPs to determine which patterns do expert pattern users consider useful or not useful for software development and maintenance. They found that only three

* Corresponding author. Tel.: +966 13 8601150.
E-mail addresses: elish@kfupm.edu.sa (M.O. Elish), mawal.mohammed@yahoo.com (M.A. Mohammed).

http://dx.doi.org/10.1016/j.infsof.2015.05.006
0950-5849/© 2015 Elsevier B.V. All rights reserved.

patterns (Observer, Composite, and Abstract Factory) were widely regarded as valuable. Zhang and Budgen [50] also conducted a mapping study to determine the scale and extent to which empirical studies have been undertaken to evaluate the effectiveness of DPs. They concluded that DPs have been subjected to limited empirical evaluation and that more empirical evidence is very much needed. The need for further investigations on the impact of DPs on different software quality attributes are justifiable by the following reasons: (i) there is no consensus among the conducted studies in the literature, as discussed in the following section, on the impact of DPs on the different quality attributes; (ii) limited number of quality attributes have been addressed; (iii) not all DPs have been addressed; and (iv) not all levels (i.e., design level, category level, motif level and role level) have been evaluated.

One of the common arguments for the applications of DPs often relate to reducing the number of software faults [23,47]. The fault density of an object-oriented class is a measure of the number of confirmed and detected faults in the class divided by its size. Class size is usually positively correlated to the number of faults, which known as the confounding effect of class size [14]. Size measures are often used to normalize fault counts when evaluating quality, as in measures of fault density [30]. The main objective of this study is to quantitatively measure and compare the fault density of design motifs in object-oriented systems. For this purpose, we conducted an empirical study that:

- Measures and compares the fault density of participant versus non-participant classes in the design motifs.
- Measures and compares the fault density of participant classes across the different categories of design motifs in the GoF's book (creational, structural, and behavioral).
- Measures and compares the fault density of participant classes across design motifs.
- Measures and compares the fault density of participant classes across the different roles in each design motif.

Quantitative analysis of fault density in design motifs provides software developers with valuable knowledge of which motifs require special attention in their implementation and testing. This knowledge serves as recommendations and guidelines for software designers to effectively apply design motifs which, in turns, help in producing better designs. Moreover, software testers can utilize this knowledge to focus on the potentially troublesome parts of designs that require more attention. Therefore, software testers can write more useful test cases that address real design issues.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 discusses the empirical study setup. Section 4 presents the results of the empirical study, which are then analyzed and discussed in Section 5. Section 6 discusses threats to validity. Finally, Section 7 provides concluding remarks and directions for future work.

2. Related work

There are many studies in the literature that have explored the relationship between DPs and software quality attributes. A comprehensive literature survey was conducted by Ali and Elish [2] on the impact of DPs on quality attributes. Only four quality attributes have been investigated in the literature: maintainability, change-proneness, performance and energy consumption, and fault-proneness.

Prechelt et al. [42] conducted a controlled experiment to study the relationship between DPs and maintainability. They found that the use of DPs improves software maintainability. Vokac et al. [48] conducted a replication of this experiment. They found that different patterns have different impact on maintainability. Another four replications for this experiment were conducted in several universities [42]. In each replication, different findings on the impact of DPs on maintainability were reported: negative impacts [27,31,41] and no impact [36]. Garzas et al. [16] investigated the impact of three different DPs on maintainability as well. In their experiment, maintainability was measured in terms of understandability and modifiability. They found that DPs make design diagrams more difficult to understand and consequently require more time to modify. Finally, Hegedus et al. [25] performed a case study on a software system to evaluate the impact of DPs on maintainability. The reported findings show an improvement in maintainability with the use of DPs.

Aversano et al. [6] conducted a case study to address the impact of DPs on change-proneness. They found that the impact of DPs on change-proneness depends on the role of DPs in the functionality of the system (i.e., if DPs are involved in the implementation of major functionalities of the system, they will be subject to more changes). Bieman et al. [8] conducted another study to evaluate the impact of DPs on change-proneness at the class-level. They found that participant classes in design motifs are more change-prone than non-participant classes. Gatrell et al. [18] conducted a case study to evaluate the change-proneness of participant versus non-participant classes as well. They found that some DPs are associated with more changes than others. Posnett et al. [40] studied the influence of DPs roles on change-proneness. They found that the observed associations between change-proneness and the roles in patterns might be due to the sizes of the classes playing those roles. Penta et al. [39] reported an empirical study that investigated the relationship between DPs roles and the frequency/kind of changes. The results confirmed the intuitive behavior about changeability of many roles. Khomh et al. [28] conducted an empirical descriptive and analytic study of classes playing zero, one, or two roles in six different DPs. They found a significant increase in many internal metric values for classes playing two roles. They also found a significant increase in the frequencies and the number of changes of classes playing two roles.

Rudzki [43] conducted a study to evaluate the impact of two DPs, Command and Façade, on performance. He found that Façade performed better than Command. Afacan [1] performed an experiment to study the impact of State design pattern on memory usage and execution time. He found that a design with the State pattern consumes more resources than a design without DPs. However, the DP solution leads to clearer system architecture that can help improve other quality attributes. Some studies have investigated the energy consumption of DPs [9,33,44]. Bunse and Stiemer [9] presented a case study that examined the impact of DPs application onto a systems energy consumption. They found that the Decorator pattern has a negative impact on the energy needs of an app. Sahin et al. [44] presented a preliminary empirical study that investigates the impacts on energy usage of applying DPs. They found that applying DPs can both increase and decrease the amount of energy used by an application, and also DPs within a category do not impact energy usage in similar ways. Litke et al. [33] observed that the use of DPs does not necessarily impose a significant penalty on power consumption.

The relationship between DPs and faults has been investigated in three works. Vokac [47] investigated fault frequency (normalized number of faults over time) in DPs. Five patterns were investigated in his study: Singleton, Template Method, Decorator, Observer and Factory Method. He found that the DPs that are associated with larger structures, such as Observer and Singleton, are subject to more faults whereas Factory Method is loosely coupled