EVALUATING THE REUSABILITY OF SYSTEMS WITH SHUFFLER DESIGN PATTERN

G PRIYALAKSHMI
Department of Applied Mathematics and Computaional Sciences, PSG College of Technology, Coimbatore
priya_venky2001@yahoo.co.in

R NADARAJAN
Department of Applied Mathematics and Computaional Sciences, PSG College of Technology, Coimbatore
nadarajan_psg@yahoo.co.in

JOSEPH W. YODER
The Refactory, Inc
joe@refactory.com

S ARTHI
Department of Applied Mathematics and Computaional Sciences , PSG College of Technology, Coimbatore
arthi.sivakumar24@gmail.com

G JAYASHREE
Department of Applied Mathematics and Computaional Sciences , PSG College of Technology, Coimbatore
jayaachu21@gmail.com

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Modern software has become intricate and versatile due to the worldwide growth of new software technologies. In this regard, the evolution of software quality metrics to support software maintainability is studied. We analyze the impact of Shuffler design pattern on software quality metrics. The Shuffler design pattern provides an alternate design approach for shuffling. The impact of Shuffler design pattern on reusability of the software systems is studied using various approaches. Firstly, object-oriented metrics provide a method for empirical analysis of impact on software quality. A list of software quality metrics, which has a higher influence on software reusability and maintainability, are experimented on three gaming applications like Jigsaw, Poker and Scramble. These gaming projects are redesigned using Shuffler design pattern and a combination of other patterns and the impact on reusability is analysed. Secondly, the reusability of the black-box components of the aforementioned projects is studied. The results with high cohesion and low coupling values would help software designers in the industry to be more confident in using the Shuffler pattern along with other design patterns. In a

nutshell, the pattern helps to choose different shuffling behavior that helps the program attain improved reusability.

Keywords: Design Patterns; Software Quality; Shuffler.

1. Introduction

The primary aim of design patterns is to achieve a good object oriented design that provides reusability, maintainability and expandability of the systems. However the use of design patterns does not necessarily result in a good design as discussed in [4]. Thus in order to evaluate the impact of design patterns, an empirical analysis of the pattern on various software quality attributes has to be studied. The justification for the use of any design pattern concerns positive effect of software quality attributes. Hence an analytical study of various patterns affecting software systems is important as stated in [9]. Object oriented metrics provide theoretical results for easier observation and evaluation of various systems.

The problem of interchange of position of objects in any application addressed by the Shuffler design pattern [2] is evaluated in this paper. Real world applications of shuffling includes shuffling questions and choices in quiz apps or online tests, image shuffling in games, shuffling wallpapers in desktops, shuffling characters in password generators etc.

In this work, we evaluate the reusability of object oriented systems with an Eclipse plug-in, called Metrics. Three open source gaming projects Jigsaw puzzle[a], Poker[b] and Scramble[c] are considered for experimental evaluation. These projects are initially evaluated with the Metrics tool for object oriented design metrics, which has an impact on the reusability and maintainability of these projects. The projects are transformed into better design solutions with the help of Shuffler design pattern. The gaming systems are modelled suitably using UML class diagrams. We studied the impact of the Shuffler Design pattern on the three systems and the results are tabulated. We also performed a combined analysis of Shuffler design pattern with other related patterns. This will greatly help software designers to promote the usage of more patterns in their designs. The patterns adopted with Shuffler design pattern were Singleton and Prototype patterns. The impact of these patterns on the quality of software systems is tabulated. We also measured the reusability of the three gaming systems at component level, with Shuffler design pattern applied. The five metrics [3] discussed by Hironori Washizaki et al, fall within the confidence interval. This justifies the increase in reusability of components designed with Shuffler pattern, without the designer having the source code.

[a]http://zetcode.com/tutorials/javagamestutorial/puzzle/
[b]https://github.com/ethnt/poker
[c]https://github.com/lisalisadong/cs-046/tree/master/problem_sets/ps5/wordScramble

2. Related Work

This section of the paper elucidates previous scientific research related to design patterns. In 1977, Christopher Alexander described the repeated occurrence of problems in our environment and suggested to devise a common solution which can be used for similar problems. In the year 1994, Gamma et al. offered the idea of design patterns, since then there has been a substantial increase in the use of design patterns. There are significant numbers of research papers discussing the impact of design patterns on quality. The probability that a class can be reused is a key quality characteristic in object-oriented design. After surveying the literature, Apostolos Ampatzoglou et al. [10] have assessed the fundamental quality attributes that are recorded to be decisive concerning the reusability of a system. The quality characteristics which they listed were Cohesion, Coupling, Messaging, Size, Inheritance, and Complexity. In 2012, Apostolos with his team [9] proposed a method to probe designs where design patterns are implemented and compare them with designs without patterns.

In their paper [1], Ivana Turnu et al. have analysed various releases of the open source Eclipse and NetBeans software systems, calculating the entropy of Response for Class and Coupling between Object Classes for every release analysed. They have demonstrated a very high interdependence between the entropy of CBO and RFC and the number of bugs for Eclipse and NetBeans. Peter Wendorff has assessed design patterns during software reengineering [7] and has justified that the incorrect usage of patterns can possibly fail. Ronald Jabangwe et al. have made an elaborate Systematic Literature Review [12] to link the object oriented measures and external quality attributes. They summarized their work by concluding that metrics that quantify complexity, cohesion, coupling and size can be useful indicators for reliability and maintainability.

In the paper [13], Jehad Al Dallal and his co-author Sandro Morasca have proved empirically that the reuse-proneness of classes can be foreseen and also enhanced by regulating their three internal quality attributes, which are size, cohesion and coupling. The author Jehad Al Dallal again in 2011 [6], proposed a cohesion metric and method to measure the discriminative power of class cohesion metrics. Foutse Khomh et al in 2008 [4] have illustrated that design patterns do not consistently boost the quality of the software. Hence they justify the demand for more studies to gauge the impact of the object oriented principles on the quality of systems. Jagdish Bansiya and Carl G. Davis have constructed a hierarchical model QMOOD [5] to inspect the quality of software products and implemented a software quality tool QMOOD++ to easily evaluate real-time projects.

3. Shuffler Design Pattern

Shuffler, a behavioural design pattern provides solution to the common problem of jumble or interchange of the objects position in any application. Shuffle method is employed for the rearrange of the positions of various objects as stated in [2]. The

pattern provides flexibility for the developer to employ various shuffling algorithms based on the application requirements. In case of complex applications the use of Shuffler design pattern is significant. If the behavior of algorithm changes at run time then Shuffler would be a special case of application of Strategy design pattern. However, Shuffler differs from strategy in applications which involve partial shuffling of objects. The influence of the pattern on reusability, a key software quality attribute is studied in the following sections. The class diagram of shuffler design pattern is as shown in Figure 1.

4. Results

This section of the paper presents the results of the assessments, according to three perspectives. The first section presents the analysis of metrics on the three open source applications with only Shuffler pattern. The second part provides the research outcomes with the applications redesigned with both Shuffler pattern and the related patterns, Singleton and Prototype. The third section presents the results of component based analysis on the aforementioned applications.

4.1. *Analysis of metrics - Shuffler pattern*

The open source gaming applications such as Poker, Jigsaw puzzle and Scramble were re-designed and implemented with Shuffler design pattern. Poker is a gambling card game in which the dealer distributes the shuffled cards to the players. This game is designed using object-oriented paradigm and implemented in Java. The same game is rejuvenated using Shuffler Design Pattern. Jigsaw Puzzle is a tiling

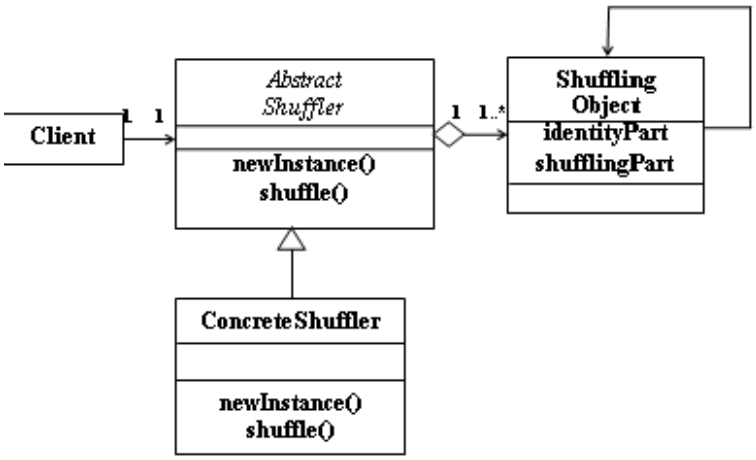

Fig. 1. Class diagram for Shuffler design pattern

puzzle game that displays shuffled pieces of an image. Scramble is a word game that jumbles the letters of a word.

For a clear visualization of the Jigsaw game design, the class diagrams of the Jigsaw Puzzle before and after applying Shuffler Pattern is shown in Figure 2, 3 respectively.

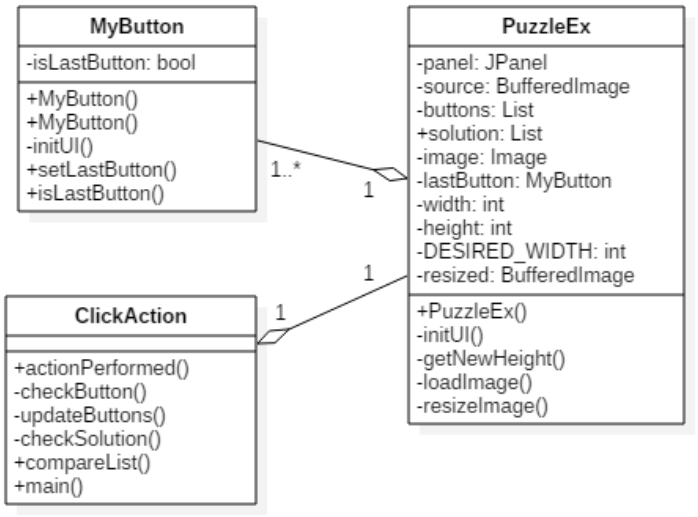

Fig. 2. Class diagram of Jigsaw game

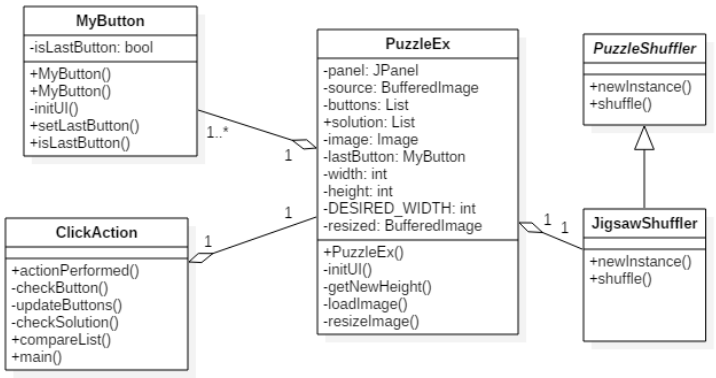

Fig. 3. Re-designed Jigsaw game with Shuffler design pattern

The quality of these software projects are assessed by using an Eclipse Plug-in, named Metrics. The impact of Shuffler Design Pattern on the above mentioned projects are recorded. The results of metric analysis on Jigsaw Puzzle, Poker and Scramble are tabulated in Table [1], Table [2] and Table [3] respectively.

	Before Pattern				After Pattern			
Mean	Total	Mean	Std. Dev.	Max	Total	Mean	Std. Dev.	Max
Depth of Inheritance Tree		4.667	1.886	6		3.4	2.154	6
McCabe Cyclomatic Complexity		1.688	1.685	7		1.55	1.532	7
Total Lines of Code	232				251			
Lack of Cohesion of Methods		0.267	0.377	0.8		0.16	0.32	0.8

Table 1. Object Oriented Metrics for Jigsaw Puzzle

	Before Pattern				After Pattern			
Mean	Total	Mean	Std. Dev.	Max	Total	Mean	Std. Dev.	Max
Depth of Inheritance Tree		1.75	0.968	3		1.7	0.9	3
McCabe Cyclomatic Complexity		2.108	2.667	18		2.059	2.617	18
Total Lines of Code	509				524			
Lack of Cohesion of Methods		0.292	0.321	0.888		0.234	0.31	0.888

Table 2. Object Oriented Metrics for Poker

	Before Pattern				After Pattern			
Mean	Total	Mean	Std. Dev.	Max	Total	Mean	Std. Dev.	Max
Depth of Inheritance Tree		1	0	1		1.25	0.433	2
McCabe Cyclomatic Complexity		1.333	0.471	2		1.143	0.35	2
Total Lines of Code	35				53			
Lack of Cohesion of Methods		0	0	0		0	0	0

Table 3. Object Oriented Metrics for Scramble

Depth of Inheritance (DIT) is a measure of inheritance of an object oriented model. Higher DIT values indicate increased reusability and complexity of the design whereas a lower value specifies less complexity of the object oriented design. A program that is likely difficult to understand has a greater McCabe number. The smaller differences in the LOC values for re-designed projects signify minimum effort required for the re-design. The assessment of abstraction for a class is given by LCOM. A low LCOM value signifies high cohesion.

A closer observation of the above tables indicates Shuffler has improved the reusability factor of Scramble project and also provides ease of understanding for Poker and Jigsaw games.

4.1.1. *Analysis of metrics Shuffler combined with related patterns*

A combination of various related design patterns were applied to the existing projects and the impact of Shuffler design pattern is studied. The Jigsaw project was implemented with Singleton pattern for the creation of Jigsaw board. Prototype pattern is used in Scramble project to create instances of the letters to be shuffled. Singleton pattern was applied to Jigsaw Puzzle along with Shuffler pattern and the results are provided in Table 4. Similarly Singleton and Prototype pattern are applied to Scramble and the results are as shown in Table 5 and Table 6 respectively.

	Before Pattern				After Pattern			
Mean	Total	Mean	Std. Dev.	Max	Total	Mean	Std. Dev.	Max
Depth of Inheritance Tree		3.4	2.154	6		3.4	2.154	6
McCabe Cyclomatic Complexity		1.55	1.532	7		1.5	1.469	7
Total Lines of Code	251				256			
Lack of Cohesion of Methods		0.16	0.32	0.8		0.16	0.32	0.8

Table 4. Object Oriented Metrics for Jigsaw Puzzle with Singleton

	Before Pattern				After Pattern			
Mean	Total	Mean	Std. Dev.	Max	Total	Mean	Std. Dev.	Max
Depth of Inheritance Tree		1.25	0.433	2		1.25	0.433	2
McCabe Cyclomatic Complexity		1.143	0.35	2		1.143	0.35	2
Total Lines of Code	53				53			
Lack of Cohesion of Methods		0	0	0		0	0	0

Table 5. Object Oriented Metrics for Scramble with Singleton

	Before Pattern				After Pattern			
Mean	Total	Mean	Std. Dev.	Max	Total	Mean	Std. Dev.	Max
Depth of Inheritance Tree		1.25	0.433	2		1.333	0.471	2
McCabe Cyclomatic Complexity		1.143	0.35	2		1.5	1.5	6
Total Lines of Code	53				85			
Lack of Cohesion of Methods		0	0	0		0	0	0

Table 6. Object Oriented Metrics for Scramble with Prototype

We infer the decrease in complexity of Jigsaw Puzzle with minimal effort for design with related patterns. However Singleton pattern do not impact the metric values on Scramble, Prototype pattern seems to have an increase in DIT value indicating high reusability of the code. The above stated results are subjected to the

individual projects and cannot be generalized to other patterns or gaming applications.

4.2. *Reusability metrics for Components*

Realization of reusability of the components in component based software development is done using metrics suite as stated in [3]. The various reusability metrics include Existence of Meta-Information (EMI), Rate of Component Observability (RCO), Rate of Component Customizability (RCC), Self-Completeness of Components Return Value (SCCr) and Self-Completeness of Components Parameter (SCCp). For each of the gaming applications re-designed using the Shuffler design pattern , the components are evaluated for reusability based on the above mentioned metrics as shown in Figure 4,5 and 6. The values of each of the five metrics within the confidence interval specified in [3] indicate higher quality of the corresponding metric.

5. Threats to Validity

This section of the paper explores possible threats to the validity of the paper. Firstly, albeit the analysis applied to study the impact ensures precision, the results on Shuffler design pattern and related patterns like Prototype and Singleton cannot be postulated to the rest of the 23 design patterns that are outlined in [1]. Next, since the gaming applications are open source projects, the experimental results may not be relevant to black-box reuse scenarios where the developer has no access to the source code. Furthermore, the dataset constituted only of Java projects, the results cannot be generalized to other object oriented programming languages.

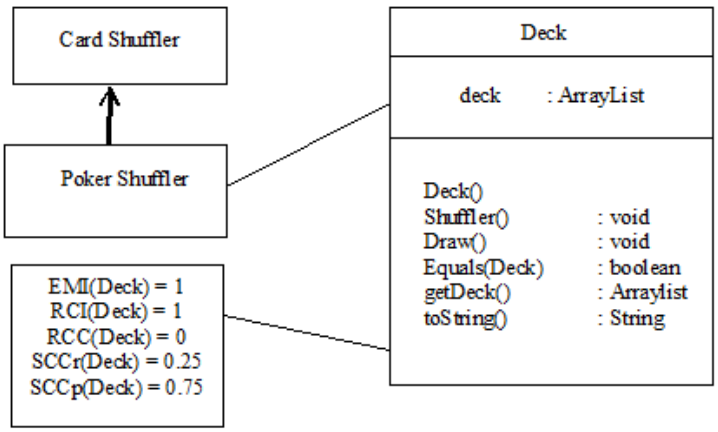

Fig. 4. Component Based Metrics for Poker

Moreover, only three projects have been considered for the study, but the statistical implications of the results imply that the results are largely okay. The results and outcome of our research confide strongly in the metrics listed in the METRICS suite, discussed in the Results section. The validity of the evaluation is confined to only these projects, thus reusability is not guaranteed for other projects. Our study does not focus on the impact on other quality attributes other than reusability.

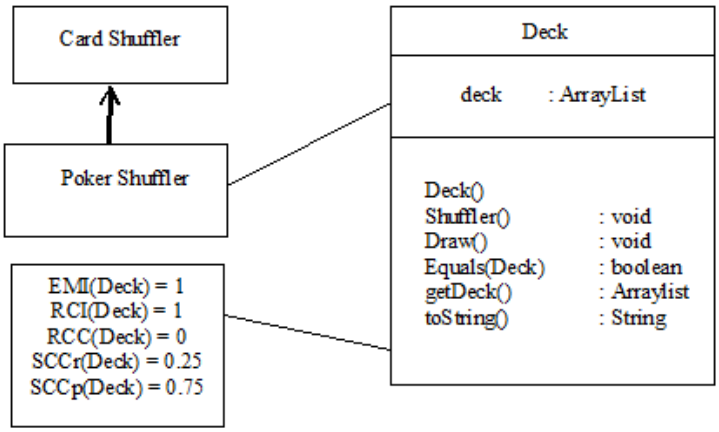

Fig. 5. Component Based Metrics for Scramble

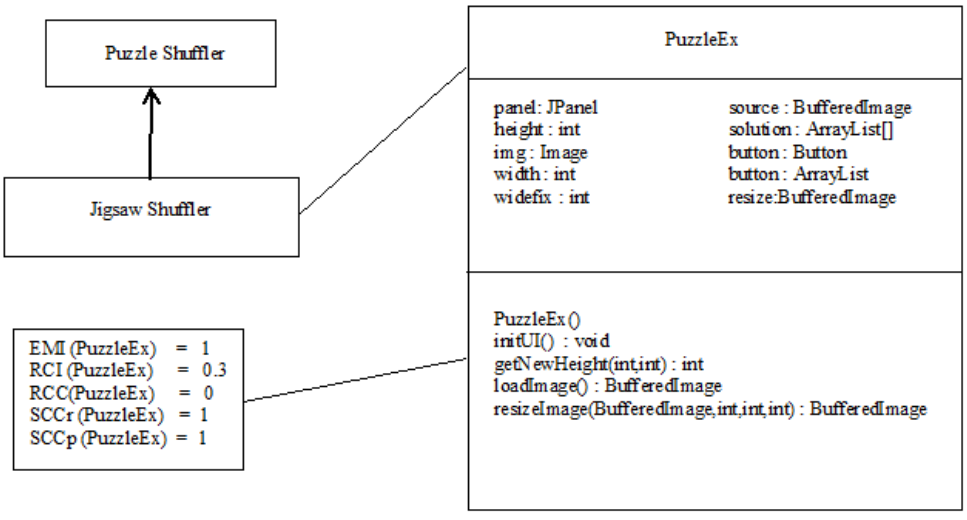

Fig. 6. Component Based Metrics for Jigsaw

6. Conclusion

This paper aims at emphasizing the effect of shuffler design pattern on quality attributes, specifically reusability. The first research effort made by the authors in the direction of design patterns was the identification of Shuffler design pattern and more importantly its known uses. After a thorough literature survey on papers related to GOF design patterns, we studied the impact of Shuffler pattern on few open source gaming application. The metrics related to the study includes Depth of Inheritance tree, Lack of Cohesion of methods, Lines of code and McCabe Cyclomatic complexity. With this study we conclude that shuffler design pattern improves reusability of the applications under study with a minimal effort for re-design of the existing projects with shuffler pattern.

The study also measures the software quality of the gaming applications with Shuffler and related patterns like Singleton and Prototype to ensure the compatibility of the shuffler pattern with other related patterns on the implementation perspective. The result observed ensures improved code reusability and high cohesion on the gaming applications.

In addition to the conventional metrics, this study also focuses on reusability metric to measure the reuse of the black box components of the three gaming applications. The advantage of this analysis is it would be helpful to designers when the source code of components is not available. The CBD analysis carried out for the projects support our claim of improved software reusability.

References

[1] Ivana Turnu, Giulio Concas, Michele Marchesi, Roberto Tonelli, Entropy of some CK metrics to assess object-oriented software quality, International Journal of Software Engineering and Knowledge Engineering, 2013.
[2] G Priyalakshmi, R Nadarajan, S Anandhi, Software Reuse with Shuffler Design Pattern, SEAT, 2016.
[3] Hironori Washizaki, Hirokazu Yamamoto, Yoshiaki Fukazawa, A Metrics Suite for Measuring Reusability of Software Components, IEEE Xplore, 2003.
[4] Foutse Khomh, Yann-Gael Gueheneuc, Do Design Patterns Impact Software Quality Positively?, IEEE Xplore, 2008.
[5] Jagdish Bansiya, Carl G. Davis A Hierarchical Model for Object-Oriented Design Quality Assessment, IEEE Transactions on Software Engineering, 2002.
[6] Jehad Al Dallal, Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics, IEEE Transactions on Software Engineering, 2011.
[7] Peter Wendorff Assessment of design patterns during software reengineering: lessons learned from a large commercial project, IEEE Xplore, 2002.
[8] Apostolos Ampatzoglou, Sofia Charalampidou, Ioannis Stamelos, Research state of the art on GoF design patterns a mapping study, Journal of Systems and Software, 2013.
[9] Apostolos Ampatzoglou, Georgia Frantzeskou, Ioannis Stamelos, A methodology to assess the impact of design patterns on software quality, Information and Software Technology, 2012.
[10] Apostolos Ampatzoglou, Apostolos Kritikos, George Kakarontzas, Ioannis Stamelos,

An empirical investigation on the reusability of design patterns and software packages, Journal of Systems and Software, 2011.

[11] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design patterns: elements of reusable object-oriented software ACM Digital Library, 1995.

[12] Ronald Jabangwe, Jrgen Brstler, Darja Smite, Claes Wohlin, Empirical Evidence on the Link between Object-Oriented Measures and External Quality Attributes: A Systematic Literature Review, Empirical Software Engineering, 2015.

[13] Jehad Al Dallal, Sandro Morasca, Predicting object-oriented class reuse-proneness using internal quality attributes, Empirical Software Engineering, 2014.