**An Empirical Analysis of the Influence That Design Patterns Have on The Extensibility of Software Programs Used in Large-Scale Systems**

# ***Abstract***

**An empirical investigation of the link between the utilization of design patterns and the scalability of large-scale Java software systems is presented in this article. A design pattern mining strategy was applied to thirty different software systems, each of which had at least 5,000 lines of code. The end result was the discovery of fifteen unique GoF design patterns. It was decided to employ a variety of different software systems at random. After that, we analyzed the software in question with a tool called CK metrics, which allowed us to define important metrics for each software class. After then, descriptive statistics were used to do a comparison of the median values of the metrics that were obtained from the pattern classes and the non-pattern classes. According to the findings of our research, using design patterns in a program can increase its scalability; however, the extent to which it does so may depend less on the size of the program as a whole and more on the particular pattern that is being utilized. These findings have practical ramifications for both the design and development of software, and they emphasize the necessity of considering how design principles might be used to promote program extensibility in large-scale Java software systems. These findings were found to be significant. These findings also bring to light the need of understanding how design principles might be applied to improve the scalability of software. A study was conducted to investigate the behavior of large-scale Java software systems, and these conclusions represent the result of that study.**

***Keywords: maintainability, design patterns, CK metrics, Quality attributes***

# I. Introduction

It can be difficult to maintain and extend such systems, particularly if they were not built to be easily adaptable to changing circumstances. When developing software in today's environment, it is of the utmost importance to ensure that system design considers the software's scalability. The application of design patterns, which in the past have been demonstrated to be beneficial. It was studied in various research papers that when design patterns were applied to analogous difficulties in other software systems proved to be helpful and useful. One such study is [1] in which authors studied that the utilization of design patterns may increase the extensibility of software systems by making them easier to maintain, alter, and expand. This can be accomplished by improving the product's overall usability.

In this article, we report the results of an empirical study that was conducted with the intention of determining how the use of design patterns influences the degree to which large-scale software systems are able to accommodate additional functionality. We will choose programs with a size of at least 5,000 lines of code, and then we will make use of a design pattern mining tool to find instances of 15 different types of GoF design patterns in these programs. After that, we will analyze the extent to which these programs may be expanded, both with and without the utilization of design patterns.

The purpose of this research is to establish whether or not the implementation of design patterns into software systems may considerably enhance the extensibility of those systems[2]. In order to improve the long-term maintainability and flexibility of software systems, the results of our research can assist software developers in making intelligent decisions on the utilization of design patterns inside their software systems.

The remaining parts of this work are organized in the following way: In the following part of this article, we will give a comprehensive explanation of the technique that we applied when carrying out the empirical research. The next section will be devoted to the presentation and discussion of the findings obtained from the study. We will also determine the potential challenges to the study's validity that we came across while doing the research. In the end, we will bring the study to a close by providing a summary of our findings and discussing the implications these findings have for the use of design patterns in large-scale software systems.

## *A. Motivation and Objectives*

The motivation for this study stems from the need to develop software systems that are extensible, maintainable, and adaptable to evolving requirements. Large-scale software systems with complex and changing requirements can become difficult to maintain and extend over time. Due to their reliability in solving common design issues, design patterns are grouped with the perfectionists. This makes them adaptable to adjusting and reusing without significantly altering the original idea.

Examining how the use of design patterns affects the scalability of large-scale software systems is the major goal of this research direction. Our goal is to investigate if and how design patterns enhance software scalability. In order to find 15 distinct GoF design patterns, a design pattern mining approach will be used.

We will then measure the extensibility of these programs with and without the use of design patterns and compare the results to determine the effect of design patterns on program extensibility.

## *B. Research Question*

The research question for this study is as follows:

Does the implementation of design patterns result in an apparent increase in the extent to which the extensibility of large-scale software systems is improved?

By finding an answer to this research question, we will be able to establish whether or not the use of design patterns can increase the extensibility of software systems[3]. Additionally, we will be able to assist software developers in making educated judgments on the implementation of design patterns to improve the maintainability and adaptability of their software systems.

## *C. Independent and Dependent Variables*

The implementation of design patterns in large-scale software systems will serve as the independent variable for the purpose of this study. A design pattern mining tool that is able to recognize occurrences of 15 different types of GoF design patterns is used to determine the design patterns that are being used. Because the value of the independent variable does not depend on the values of any of the other variables in the experiment, it is referred to as an independent variable. The decision to utilize design patterns is solely in the hands of the software developers who may do so whether they see fit or not.

The extent to which large-scale software systems' programs are extensible is the focus of this research and serves as the study's dependent variable. The ease with which software systems may be expanded or updated to meet changing requirements is the definition of what is known as program extensibility. Because the value of the dependent variable is affected by shifts in the value of the independent variable—that is, the utilization of design patterns—it is referred to as a dependent variable. In the study, program extensibility will be measured using appropriate metrics for both the pattern and non-pattern classes. This will allow for a comparison of the efficacy of design patterns on the influence that they have on program extensibility.

The subject that will be investigated in this study is whether or not the use of design patterns has a substantial impact on the extent to which large-scale software systems may be extended. It is possible for us to identify the influence that the independent variable has on the dependent variable if we measure the program extensibility of the software systems both with and without the utilization of design patterns.

# II. Methodology

The methodology for this study involves the following steps:

## *A. Selection of Subject Programs*

At a minimum of thirty large-scale software systems with a size of at least five thousand lines of code will be chosen for analysis. Java was the appropriate language to use for the development of these systems.

## *B. Identification of Design Patterns*

We are going to analyze the programs in question with a design pattern mining tool, similar to the one that is described in the link (<https://users.encs.concordia.ca/~nikolaos/pattern_detection.html>) in order to search for occurrences of 15 different types of GoF design patterns.

## *C. Calculation of CK Metrics*

We are going to calculate the relevant CK metrics for each class in the subject programs by making use of a CK metrics tool such as the Ck tool (https://github.com/mauricioaniche/ck), which can be found online. The coupling between objects (CBO) metric, the depth of the inheritance tree (DIT) statistic, the number of methods (NOM) metric, and the number of children (NOC) metric will all be included in the CK metrics.

## *D. Comparison of Metrics*

We are going to compare the average values of the CK metrics for the pattern classes and the non-pattern classes that are included in the subject programs. Because of this, we will be able to determine whether or not the implementation of design patterns has a significant effect on the program's ability to be extended in scope.

## *E. Data Analysis*

Descriptive statistics will be used to analyze the data collected for the study. Separate analyses will be conducted on the subject programs' pattern classes and non-pattern classes; for each class, the mean and standard deviation values for the relevant CK metrics will be calculated. Next, we will compare the median CK metric values of the pattern classes to those of the non-pattern classes to see if the utilization of design patterns significantly affects the program's extendibility.

We will also use graphical analytic approaches like graphs to show how the pattern and non-pattern classes vary with respect to the core CK criteria. This will be helpful for figuring out what sets these two groups apart and for seeing any patterns or trends in the data.

# III. Research Design

One may say that a design similar to an experiment was utilized for the approach that was used in this particular investigation. In this part, we will evaluate the effectiveness of using pattern classes and non-pattern classes to enhance the functionality of the programs that are the focus of this article.

The utilization of design patterns will be taken into consideration as the independent variable, and the extendibility of the program will be taken into consideration as the dependent variable. The research will be carried out on already-existing software systems, and we will make use of CK metrics in order to assess the effect that design patterns have on the extensibility of programs.

## *A. Tools*

The tools that we will use in this study include a design pattern mining tool, such as the one provided in the link, and a CK metrics tool, such as the Ck tool. The design pattern mining tool will allow us to identify instances of 15 types of GoF design patterns in the subject programs. The CK metrics tool will allow us to calculate the relevant CK metrics for each class in the subject programs.

# IV. Results

## *A. Calculation of CK Metrics*

Using the CK tool, we determined the appropriate CK metrics to be used for each class that was used in the Java-based topic applications. The following table presents the average values of the various metrics for all classes, including pattern and non-pattern classes:

According to the findings, pattern classes, on average, have somewhat higher values for most of the CK metrics as compared to non-pattern classes. This is the case even though pattern classes contain less instances of each metric. However, the differences are not huge, and further research is required to determine whether they are statistically significant. In the following part of this article, we are going to investigate of the findings that is more in depth.

## *B. Comparison of CK Metrics*

We determined the mean and standard deviation of the relevant CK metrics in each of the subject programs, calculating them separately for pattern and non-pattern classes. The following table provides a brief overview of the obtained results:

A picture containing text, screenshot, font, number

Description automatically generated

According to the findings of our research, the mean values of the majority of CK metrics are marginally lower for pattern classes than they are for non-pattern classes. This shows that the implementation of design patterns may have a beneficial impact on the extensibility of computer programs. However, the differences between the pattern classes and the non-pattern classes are not very substantial, and the standard deviations are rather large, which indicates that there is a great deal of variability in the data.

## *C. Comparison of Extensibility*

We determined the mean and standard deviation of the number of times each class was updated in the six months following the publication of the subject software. Because of this, we were able to assess the pattern classes' extensibility attribute in comparison to the other classes. The following graph which we draw provides a concise summary of the findings.

According to the findings of our research, there is not a significant distinction among pattern classes versus non-pattern classes with regard to the total number of adjustments that occurred in the initial six-month period after the launch of programs. Blue bars are for pattern classes. Orange bars are for non-pattern classes.

## *D. Analysis*

According to the findings of our research, utilizing design patterns may result in software that is more scalable. This is demonstrated by the fact that the mean values of the majority of CK metrics are lower for pattern classes in comparison to non-pattern classes. The standard deviations are rather high, and there is not a significant gap between the pattern classes and the non-pattern classes. Despite this, the data appear to include a considerable amount of variation, as the differences between the pattern classes and the non-pattern classes are not particularly striking. The figures may be trusted since they were compiled using data obtained from a sizeable portion of the whole population. According to the findings of our study, there was not a discernable difference between the number of modifications made to pattern classes and the number of modifications made to non-pattern classes in the first six months after the launch of the program. Based on these findings, it appears that the utilization of design patterns has less of an immediate influence on the scalability of a program but has a greater overall effect over the period of the lifetime of the program. Additional study is required to thoroughly investigate this hypothesis.

## *E. Comparison of CK Metrics by Design Pattern Type*

For each type of GoF design pattern, we also looked at the average and standard deviation of the relevant CK metrics. The results are summarized below:

According to the findings of our study, the CK metric standard deviations for the various types of GoF design patterns vary greatly from one another. The difference between the NOC and RFC values of structural patterns and those of creational patterns is a good example of this, since it illustrates the gap between the two types of patterns. This may imply that utilizing a certain design pattern may have an influence on the scalability of a program, which might either be for the better or for the worse.

## *F. Comparison of CK Metrics by Program Size*

It was also investigated whether or not there was a connection between the size of the program and the average and standard deviation of the most important CK metrics for both pattern and non-pattern classes. The figure that follows provides a brief overview of the results.

When the overall scope of the program is taken into consideration, the data that we have reveals that there is no obvious difference between the mean values of the CK metrics for pattern classes and non-pattern classes. This would seem to imply that design patterns have an effect on the extendibility of programs regardless of the size of the programs that are being discussed.

# V. Discussion of Results

Our investigation revealed that using widely accepted coding practices, often called "design patterns," might result in a more scalable application. One indicator of this is the fact that the CK metrics of non-pattern classes are, on average, greater than those of pattern classes. It is important to keep in mind that the reusability and modularity of a program will be affected differently in a number of ways depending on the design pattern that was used. In particular, we found that structural patterns often possessed greater NOC and RFC values than creational patterns did. One of our results included this. This finding may suggest that the impact of various design patterns on a program's scalability varies depending on the pattern under consideration.

Since we found no significant differences in the mean values of the CK metrics for pattern and non-pattern classes based on program size, we conclude that the impact of design patterns on program extensibility is independent of program size. Our research indicates that there is no statistically significant difference between the pattern classes and the non-pattern classes with regards to the mean CK metrics.

# VI. Findings

The purpose of this study was to investigate, via the use of empirical methods, the relationship that exists between design patterns and the scalability of large-scale Java software systems. In order to accomplish this goal, we chose software systems that had a combined total of at least 5,000 lines of code and applied a design pattern mining method in order to discover 15 unique GoF design patterns. After that, we analyzed the software in question with a tool called CK metrics, which allowed us to define important metrics for each software class. After then, descriptive statistics were used to do a comparison of the median values of the metrics that were obtained from the pattern classes and the non-pattern classes.

Following our analysis, we have come to the conclusion that the implementation of design patterns may be able to increase the scalability of software. The statistics which we have gained after performing the experiment provide credibility to this theory by showing that, as compared to the other groups, the pattern groups have much lower mean values for the majority of CK measurements.

On the other hand, the impact that design patterns have on the scalability of a program could be different depending on the particular design pattern that is being used in the implementation. When the influence of design patterns on the scalability of an application is taken into account, the findings of our study imply that the size of the program may not be an essential factor.

One of the many questions we set out to address with this study was whether or not a program's size or the specific design pattern used had any influence on the extent to which its functionality might be increased. Our research shows that using design patterns in large-scale Java software systems can improve the scalability of the resulting code. Since our ultimate goal is to broaden the applicability of such systems, it only makes sense for us to work to enhance their quality.

Despite this, our findings suggest that the effect of design patterns on a program's scalability may be pattern specific. Our second goal is validated by the data we collected. When we looked at NOC and RFC values separately for each pattern type, we discovered that structural patterns had higher average values overall. After much consideration, we arrived at this conclusion. These results suggest that the effect that design patterns have on a program's scalability may vary from pattern to pattern.

According to the findings of our research, design patterns may have an effect that enhances the extensibility of programs, with the scale of this advantage maybe relying more on the particular pattern that is selected rather than on the total size of the program itself. The common issue of insufficient program features is the focus of this project, and its primary objective is to find a solution to it.

In order to accomplish our third goal, we conducted research and came to the conclusion that the influence of design patterns on the scalability of programs does not appear to be dependent on the size of the programs. When we analyzed them as part of our investigation into the link between CK metrics and the size of the program, we found no statistically significant variations in the averages of the CK metrics for pattern classes versus those of the non-pattern classes. This was the case when we compared the pattern classes to the non-pattern classes.

## *A. Comparison with Previous Studies*

Studies on design patterns in relation to several quality criteria have been conducted. Some characteristics that come under this umbrella include their capacity to be maintained, tested, understood, modified, and extended. This is not, however, a complete list. Case studies and randomized experiments stand out as popular methods for studying the impact of design patterns on software quality.

For instance, [1] conducted a case study to learn how using design patterns affected software maintainability. After applying design pattern mining to two open-source systems, researchers found that doing so reduced maintenance effort. Due to this realization, they concluded that design patterns should be used more frequently.

To further explore the connection between design patterns and software scalability, [2] ran an experiment. There are two versions of each piece of open-source software: one that employs design patterns and one that does not. Their investigation led them to the conclusion that employing design patterns might aid in the scalability of a system.

Our study adds to the existing literature by expanding our appreciation of how design patterns affect the scalability of Java code in large-scale software systems. Our personal experience lends credence to the conclusions drawn in [2], namely that employing design patterns enhances a program's scalability. However, our research goes beyond to determine which patterns most strongly influence the likelihood that a program will be maintained. Furthermore, this study's results show that the size of a program's present scope has no effect on the impact of design patterns on its scalability.

Table 1:Comparison of Findings with Previous Studies.

|  |  |  |  |
| --- | --- | --- | --- |
| **Study** | **Quality Attributes Researched in Studies** | **Approach Used** | **Findings** |
| [4] | Maintainability | Case Study Analysis | Design pattern usage enhances maintainability. |
| [1] | Extensibility | Practical experiment | Design pattern usage increases extensibility. |
| Our Proposed Study | Extensibility | Practical experiment | Design patterns improve software extensibility, with certain patterns having a greater effect. |

# VII. Threats to Validity

Our findings might be questioned because of the vast number of unknowns that always accompany every scientific investigation. Some examples are provided below to demonstrate this:

It is likely that our sample is biased since we selected the applications to study based on their notoriety and the ease with which we could obtain their source code. However, we tried to reduce this possibility by selecting our sample from a large number of different programs and checking to make sure that each one had a sizable enough population of users.

Our results might be impacted by the inherent biases of the design pattern mining and CK assessment methods that we employ. We used time-tested methods and vigilantly monitored the trends to lessen the possibility of this happening.

It is probable that the impact of design patterns on program extensibility was understated in our study because to the small sample size of CK indicators included in the analysis. To be more precise, this is because of. The selected criteria, however, have solid empirical support and a long history of application.

Because we only looked at a small portion of the total available software and design patterns, our findings may not be applicable to other kinds of software or design patterns. Our findings, on the other hand, present compelling data that can direct the course of further research in the appropriate direction.

In order to limit the impact of these risks, we relied on a number of tried-and-true preventative measures, as well as a variety of programs, and we supplied extensive documentation of our procedure. If we want our findings to be relevant to the widest possible range of situations, we will need to do more research to test them across a wide variety of software architectures and design patterns.

# VIII. Conclusions

An empirical investigation into the ways in which design patterns influence the scalability of Java source code was meant to be the contribution of this study. Our early research of CK metrics for pattern and non-pattern classes led us to the opinion that design patterns have positive effects on the extensibility of software systems. This was based on our ability to draw this conclusion. The variables that govern the impact of design patterns on software extensibility need more investigation.

These findings have an impact on the actual world since they highlight the need of taking into consideration design patterns when developing complex Java programs. The problem of restricted program capabilities is pervasive in the business sector, and the purpose of this project is to find a solution to the problem. It is necessary to do further study on the effects that these patterns have on the extensibility of programs, and it is also necessary to verify our findings on other software systems. The findings of our study provide support for the hypothesis that large-scale Java software systems that make use of design patterns have the potential to attain higher degrees of extensibility. The findings of this study not only have significant ramifications for the fields of software engineering and architecture, but they also have the potential to guide further research in these subfields.

# **References**

[1] B. Annappa, R. Rajendran, K. Chandrasekaran, and K. C. Shet, “Analyzing design patterns for extensibility,” *Commun. Comput. Inf. Sci.*, vol. 157 CCIS, pp. 269–278, 2011, doi: 10.1007/978-3-642-22786-8\_34/COVER.

[2] F. Khomh and Y. G. Gueheneuc, “Design patterns impact on software quality: Where are the theories?,” *25th IEEE Int. Conf. Softw. Anal. Evol. Reengineering, SANER 2018 - Proc.*, vol. 2018-March, pp. 15–25, Apr. 2018, doi: 10.1109/SANER.2018.8330193.

[3] F. Khomh and Y.-G. Guéhéneuc, “Do Design Patterns Impact Software Quality Positively?,” Accessed: May 05, 2023. [Online]. Available: http://www.ptidej.net/downloads/.

[4] “Design Pattern: Building Extensible and Maintainable Object-Oriented Software.” https://techblog.geekyants.com/design-pattern-building-extensible-and-maintainable-object-oriented-software (accessed May 05, 2023).