Correlation Between the Frequent Use of Gang-of-Four Design Patterns and Structural Complexity

Shahid Hussain, Jacky Keung, Arif Ali Khan, Kwabena Ebo Bennin
Department of Computer Science, City University of Hong Kong
Shussain7-c@my.cityu.edu.hk, jacky.keung@cityu.edu.hk, aakhan2-c@my.cityu.edu.hk, kebennin2-c@my.cityu.edu.hk

ABSTRACT

The structural complexity of design components (e.g. Classes) is proportional to design quality at the system level and is quantified via the object-oriented metrics. The frequent use of design patterns causes of too much abstraction and can increase the structural complexity of design components. Though, in our previous work, we have empirically investigated the impact of use intensity of design pattern on the system level quality attributes. However, the empirical investigation of the effect of usage of design patterns on the design properties is still required. In this regard, we conduct an empirical study and perform a case study which includes the analysis 1) the existence of a correlation between design pattern usage and design metrics, 2) the confounding effect of system size (number of classes) on the correlation, and 3) how the change in number of employed design pattern instances affects the structural complexity in the subsequent releases of a system. The result of this study suggests that structural complexity associated with aggregation, coupling, functional abstraction design properties has a significant relationship with the employed instances of Template, Adapter-Command, Singleton, and Factory Method design patterns.

CCS Concepts
• **General and Reference→ Cross-computing tools and techniques→***Empirical studies; Design; Metrics;*

Keywords
Design Patterns; Complexity; Metrics; Object Oriented; Class; Coupling.

1. INTRODUCTION

The structure of an object-oriented application includes a set of design components named classes and packages, besides their design properties such as inheritance, abstraction, aggregation, and composition. In software engineering, the term pattern is used in different software development phases (such as in requirement and design phase) and with different application domains (such as enterprise applications) in order to refer a solution of a commonly occurring problem. In the domain of object-oriented software development, Gamma, Helm, Johnson, and Vlissides (also known as Gang-of-Four or GoF) introduce a catalog of 23 design patterns that describes simple and well-designed solutions to specific problems in object-oriented design [6]. The usage of design patterns can aid developers to provide reuseable solutions in order to reduce the system level complexity and increase the maintainability of an application. However, developers, who have less awareness with the use of design patterns, believe that the use of design patterns can increase the complexity of an application. There are three main dimensions to compute the compute the complexity of an evolved system such as 1) Structural, 2) Dynamic, and 3) Organizational Complexity [16, 17, 18]. Structural complexity aid to characterize the system's architecture. Dynamic complexity refers to complexity related to the dynamical behavior of the systems. Moreover, the organizational complexity refers to the complexity related to system development process and organization structure of the developer's team.

In this study, we addressed developer's concerns, and empirically investigate the effect of the GoF design patterns on structural complexity of an application through the correlation exist between the frequent use of design patterns and design properties used to assess the structural complexity of design components. For example, coupling, inheritance, polymorphism, messaging and composition that can help to examine the structural complexity of design components such as classes. Numerous design metrics have been theoretically and empirically investigated to quantify the design properties and system level quality attributes. Though in our previous study, we have investigated the effect of design pattern usage on the system level quality attributes [15]. However, the empirical investigation of structural level complexity at system level is still required. In this regard, we consider QMOOD design metric suit [5], Direct Class Coupling (DCC), Measure of Aggregation (MOA), Average Number of Ancestor (ANA), Measure of Functional Abstraction (MFA) are used to quantify the Coupling, Composition, Abstraction and inheritance design properties. Subsequently, from Chidamber and Kemerer (CK) suit [3], Coupling Between Object (CBO), Response For a Class (RFC) and Depth of Inheritance Tree (DIT) are used to quantify the coupling, messaging and inheritance design properties.

978-1-5386-3681-7/17 $31.00 © 2017 IEEE
DOI 10.1109/APSEC.2017.25

The aim of our study is to perform a case study which includes three analyses in order to empirically investigate the effect of usage of design patterns on the system level structural complexity of a system. Moreover, we also investigate the two factors, which can affect the correlation (between employed design pattern instances and design properties) that is confounding effect of system size (number of classes) and change effect in subsequent releases of a system. In this case study, the cases and unit of analysis are the Eclipse Java-based open source software projects which are retrieved from the Qualitas.Class corpus in compiled form [13]. In order to achieve our research objective and respond to our research questions, we perform data analysis using Spearman correlation, descriptive statistics, non-parametric tests for hypothesis testing, 2-D, and 3-D area charts.

The rest of the paper is structured in the six sections: In section 2, we present the related work which highlights the importance and motivation of our research objective. In section 3, we provide a brief introduction CK and QMOOD design metrics, consider in our study and help to quantify the design properties. In section 4, we present the structure of pro-posed case study, which includes subsections of research objective, research questions, cases, and unit of analysis, used tools and data analysis process. In section 5, we present and discuss the results in order to respond our research questions. In section 6, we present threats to validity, and finally, in section 7, we present the conclusion and future work.

2. RELATED WORK

An object-oriented application with less complexity is easy to maintain. We summarized related work on the base of correlation exist between software maintainability and GoF design patterns [2, 8, 9, 10]. Recently, in their study, A. Ampatzoglou et al [1] reported the interest of the software engineering research community for Gang-of-Four (GoF) design patterns in both academia and industry. In their mapping study of more than 130 scientific papers, they reported the effect of GOF design patterns on the software quality attributes remain more important research topic as compared to pattern formulations and pattern detection. C. Zhang and D. Budgen [2] also reported the use of patterns in their mapping study and provide a framework for the maintenance. Subsequently, the authors recommend that researchers should use case studies that should be focused on the key patterns and their use impact on the quality attributes. M. Vokac et al [8] replicates the experimental study of L. Ratchet et al [9], and investigated the usefulness of design patterns with respect to maintenance in a design program, even though design problem is simpler than that solve by a pattern. Moreover, M. Vokac et al used real programming environment instead of pen and paper. In both studies, the authors compare the maintainability of systems with and without design patterns and use the same questionnaire to collect the data from professionals. Moreover, the authors considered the Abstract Factory, Composite, Observer, Visitor, and Decorator design patterns, and reported different results especially in the case of Visitor and Observer patterns. Finally, the experimental results of both studies suggest that the impact of design patterns is either harmful or beneficial with respect to maintenance. Moreover, the applicability of GoF design pattern depends on decision provoked by designers through their common sense.

Recently, P. Sfetsos et al [14] investigate and compare the effectiveness of employed design pattern instances in the design quality of software libraries and standalone application. While these studies have empirically investigated and reported the relationship between GoF design patterns and design quality, however, they cannot report the effect of employed instances of the GoF design patterns on the structural complexity of a system.

Therefore, we present a case study to empirically explore the correlation between the employed instances of the GoF design patterns and system level design complexity. We also investigate the factors which can affect the correlation, such as the confounding effect of system size and change in design pattern instances during the evolution of a system.

3. DESIGN METRICS TO QUANTIFY DESIGN PROPERTIES

Numerous object-oriented design metrics suits have been theoretically and empirically investigated to quantify the different design properties, in order to assess the structural complexity of design components. For example, Direct Class Coupling (DCC) metric from QMOOD, used to quantify the direct coupling of a class with other classes through attribute declaration or message passing [5]. However, in this study, we consider design metrics from two well-known metrics suits C&K and QMOOD, which have been empirically investigated to measure the structural complexity and its implication to defects [3, 4, and 5]. The list of design metrics, aid to quantify the structural complexity of design component (i.e. Class), is shown in Table 1.

Table 1. Design Properties and Design Metrics

Design Properties	Design Metrics	Metrics Suit
Abstraction	ANA	QMOOD
Class Coupling	DCC	QMOOD
Composition	MOA	QMOOD
Inheritance	MFA	QMOOD
Objects Coupling	CBO	C&K
Messaging	RFC	C&K

The brief description of design metrics (shown in the Table 1) is as follows:

- Average Number of Ancestor (ANA)

The ANA metric quantify the average number of classes from which a class can inherit information. This metric is computed by determining the number of classes including all paths from the root class of all classes in an inheritance structure.

- Direct Class Coupling (DCC)

The DCC metric count the different number of classes which are directly related by attributes declaration and message passing (parameters) in methods.

- Measure Of Aggregation (MOA)

The MOA metric quantify the extent of the part-whole relationship realized through user defined attributes. It counts the number of data declarations whose types are used in the defined classes.

- Measure of Functional Abstraction (MFA)

The MFA metric refers to the ratio of the number of methods inherited by a class to the total number of methods accessible by member methods of the class.

- Coupling Between Objects (CBO)

The CBO metric value of any specific class is equal to the number of other classes, to which specific class is coupled.

- Response For a Class (RFC)

The RFC metric value is equal to the number of methods that could be executed in response to a message received by an object of a class.

The selected metrics can quantify the different aspects of design complexity of a system. However, there are some other design metrics, which can quantify the internal complexity of classes rather than structural (i.e. External) and is related to some other design properties. For example, Number of Methods (NOM) in QMOOD and Weighted Methods per Class (WMC) in C&K metrics suit. Since, the aim of our study is to empirically investigate the effect of the GoF design patterns on the system level structural complexity of a system, consequently, we consider only six design metrics (shown in Table 1) in this study.

4. STRUCTURE OF CASE STUDY

In this paper, we accomplish a case study that is an observational empirical method, which can be used to monitor the project relevant activities in a real-life context [11]. We present this case study, according to rules suggested by P. Runeson et al [11]. In the domain of software engineering, P. Runeson et al present the design templates of two case studies named holistic and embedded case study, which are further classified as single case and multiple-case study. According to P. Runeson et al, we noted that the holistic case study is that from which we can extract only one unit of analysis from each case, while in the case of the embedded case study, multiple units of analysis can be extracted from a single case. In our study, we follow the holistic multiple-case study template and present three types of analysis in order to achieve our research objective. We perform these analyses with 51 open source projects and nine subsequent releases of a system named velocity. We used Qualitas.Class corpus to retrieve the compiled Eclipse Java open source software projects [13]. In each analysis of case study, we considered open source software projects as cases and unit of analysis. The results of the case study will aid the agile developers in order to evaluate the effect of employed instances of the GoF design patterns on the structural complexity of open source applications. Moreover, agile developers can also analyze and compare the effectiveness of the employed instance of the GoF design patterns across the open source applications with respect to system size (number of classes) and subsequent releases of a system.

The structure of this case study is described in the following subsections of research objectives and research questions, data collection and used tools, and data analysis process.

4.1 Research Objectives and Research Questions (RQs)

The primary objective of our study is to empirically investigate the effect of use intensity of GoF design pattern instances on the structural complexity of a system. We used a set of design metrics as assessor to quantify the structural complexity of classes in order to assess different design properties. The first sub-objective is to empirically investigate the existence of a correlation between the use intensity of the GoF design patterns and structural complexity assessors. Through the statement of our first sub-objective, we extract and formulate our first research question.

RQ-1. Is there any correlation between use intensity of GoF design patterns and structural complexity assessors?

Subsequently, in their study, A. Ampatzoglou and A. Chatzigeorgiou[7] reported that the GoF design pattern occurrence can improve the design properties such as complexity, cohesion and coupling, however, it is highly provoked with system size in term of a number of classes or line of code. The second sub-objective of our study is to investigate the confounding effect of system size on the correlation between use intensity of the GoF design patterns and structural complexity assessors. Through the statement of our second sub-objective, we extract and formulate our second research question.

RQ-2. What is the impact of the confounding effect of system size (# of classes) on the correlation between use intensity of the GoF design patterns and structural complexity assessors?

Moreover, in the evolution process of software systems, it has been empirically validated that employed design

pattern instances are changed and causes a modification to the design components (such as method or classes), either as participant or non-participant in the design pattern. Subsequently, the complexity level of design components varies with the change in employed instances of design patterns. The change in the incidence of design pattern instances in any release of a software system can also affect the complexity level and design quality in its subsequent releases. The third sub-objective of our study is to investigate the change in the correlation between use intensity of the GoF design patterns and structural complexity levels in subsequent releases of a system. Through the statement of our third sub-objective, we extract and formulate our third research question.

RQ-3. What is the effect of a change in the employed instances of a GoF pattern on the structural complexity levels of design components in the subsequent releases of a system?

Table 2. Structure of Dataset

Variable(s)	Alias	Description
V1	Project Name	We used this variable to record the name of project as a case or unit of analysis.
V2	Project Size	This Variable refers to a number of classes exist in a project.
V3-V14	Use Intensity of Design Patterns	We used these variables to record the use intensity of each type of design patterns, including Proxy2 as a variation of Proxy pattern. The variable V2 to V13 refer to Singleton, Factory Method, Prototype, Adapter-command, Observer, State-Strategy, Template Method, Visitor, Composite, Decorator, Proxy and its variation Proxy2 respectively. The values of these variables are normalized with system size and the employed design pattern instances, that is, each value is calculated as the fraction of the number of classes (system/project size) divided by the number of instances of corresponding design pattern. The low value of variable refers to the intensive use of the corresponding pattern in a project.
V15-V24	Metric values for Design Paroperties	We used these variables to record the metric values which help to quantify the design properties (such as cohesion, coupling, abstraction and so on) and design quality attributes. The value of each variable (from V15 to V24) of a case (an open source project) refers to an Average Value (AV) and calculated as $AV_{i,k} = \frac{\sum_{j=1}^{N} \text{Value\_of\_Case}_i\text{\_for\_Class}_j}{N}$ $\forall$ i=1 to n and k=15 to 24. The value of $AV_{i,\ k\ refers}$ to the average value of variable k in case I, such that i ε n, where n refer to the total number of cases in the dataset. The term N refers to the system size that the total number of classes for each case i.

4.2 Data Collection and Used Tools

We employed two different tools to collect data from the compiled Eclipse Java open source projects retrieved from Qualitas.class corpus[1]. The first tool is SSA (Similarity Scoring Algorithm) introduces by Tsantalis[2] et al [12] and it is based on a similarity scoring algorithm.

Table 3. Spearman Correlation between GoF design pattern and Design Metrics

Design Patterns	MOA	MFA	ANA	CBO	RFC	DCC
Singleton	0.59(0.00)	0.37(0.03)	-0.62(0.00)	0.50(0.00)	-0.38(0.03)	0.44(0.00)
Factory Method	0.50(0.00)	0.37(0.01)	-0.50(0.00)	0.54(0.00)	-0.35(0.03)	0.38(0.00)
Prototype	0.37(0.01)	-0.03 (0.93)	-0.35(0.01)	0.38(0.02)	-0.23(0.10)	0.20(0.11)
Adapter-Command	0.63(0.00)	0.28(0.03)	-0.61(0.00)	0.55(0.00)	-0.33(0.01)	0.53(0.00)
Observer	0.66(0.00)	0.37(0.02)	-0.60(0.00)	0.61(0.00)	-0.31(0.02)	0.37(0.00)
State-strategy	0.62(0.00)	0.32(0.01)	-0.6(0.00)	0.67(0.00)	-0.34(0.01)	0.47(0.00)
Template Method	0.52(0.00)	0.17(0.24)	-0.62(0.00)	0.62(0.00)	-0.15(0.29)	0.29(0.03)
Visitor	0.03(0.61)	0.06(0.51)	-0.07(0.61)	0.05(0.75)	-0.05(0.63)	0.12(0.39)
Composite	0.21(0.09)	0.09(0.51)	-0.23(0.09)	0.27(0.05)	-0.21(0.14)	0.34(0.01)
Decorator	0.45(0.00)	0.15(0.40)	-0.47(0.00)	0.48(0.00)	-0.23(0.10)	0.37(0.00)
Proxy	0.50(0.00)	0.09(0.00)	-0.43(0.00)	0.48(0.00)	-0.53(0.00)	0.43(0.00)
Proxy2	-0.03(0.84)	0.08(0.58)	0.02(0.85)	-0.01(0.92)	-0.22(0.12)	0.20(0.12)

Moreover, the SSA tool is capable of identifying automatically the occurrences of the GoF design patterns exist either in standard or in variation form. The second tool is CKJM-extend[3] which is extended version of CKJM (Chidamber and Kemerer Java Metrics) tool. Moreover,

[1]http://java.labsoft.dcc.ufmg.br/qualitas.class/download.html
[2]http://users.encs.concordia.ca/~nikolaos/pattern_detection.html
[3]http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/

this tool helps to measure the metrics of different suites such as Chidamber and Kemerer (CK), QMOOD, Tang et al and Martin suites. The values of these metrics can be used to quantify the design properties and system level quality attributes. In order to perform their function, both tool need compiled form (byte code) of open source project and can be downloaded from the web[2,3]. The structure of a created dataset is described in the Table 2.

4.3 Data Analysis

We performed three analyses using descriptive statistics, which include mean values and Spearman correlation, and graphs which include 2D line chart and 3D area charts to present monotonic relationship and effect of employed instances of design patterns on the design structural complexity. Moreover, we also perform hypothesis testing using non-parametric Friedman test in order to evaluate the design structural complexity differences among project's groups with respect to system size.

5. RESULTS AND DISCUSSION

In this section, we present the results of our case study, which are organized to respond our research questions.

5.1 Respond to RQ-1.

In order to respond to our first research question (RQ-1), we perform a non-parametric (distribution free) Spearman's correlation test to investigate the monotonic relationship between use intensity of design patterns and structural complexity assessors. The Spearman's correlation (Corr) and significant value p (represented as Corr (p) in the table) for each GoF design pattern and corresponding structural complexity assessors are shown in the Table 3.

In the Table 3, the high value with a negative sign also presents the significant correlation between a number of employed instances of design patterns and assessors. The negative sign is due to the fact that design structural complexity is quantified using relevant design metrics and use intensity of design patterns is acquired by the fraction of a number of classes and number of instances of the corresponding design pattern. From the Table 3, we can conclude the following findings:

- We observe the significant correlation (at the level $p<0.05$) between the number of employed instances of Singleton, Factory Method, Adapter-Command and Template-Method design patterns and structural complexity assessors. These design patterns are frequently used and their aggregate effect is evident at the system level.

- We observe an insignificant correlation of employed instances of proxy2 (a variant of proxy) on design structural complexity, it might be that it cannot be detected in many applications.
- We observe the differences in the significance effect of the number of employed instances of rest of design patterns on the design quality attributes, such as the number of employed instances of the Observer design pattern have a significant effect on all structural complexity assessors except ANA and RFC.

From the findings of Table 3, we can conclude that there is evidence in the support of existence of a correlation between use intensity of design patterns and structural complexity assessors; however, their correlation is highly related with the number of employed instance of design patterns and the average values of assessors (i.e. Design metrics) computed at the system level.

Table 4. Group's information

Groups	# of Cases	System Size (# of classes)
Group-1	17	<100
Group-2	17	100-1000
Group-3	17	>1000

Table 5. Descriptive Statistics of Design Metrics

Quality Attributes	Mean Values		
	Group-1	Group-2	Group-3
MOA	0.81	0.77	0.91
MFA	0.07	0.12	0.15
ANA	0.10	0.07	0.13
CBO	2.60	4.16	5.75
RFC	8.44	7.54	8.69
DCC	6.26	8.07	9.80

5.2 Respond to RQ-2.

In order to respond our RQ-2, we perform an analysis with respect to system size. We divide the cases (that is open source projects) into three groups that are Group-1, Group-2, and Group-3. For each group, we create a new dataset of the same structure defined in the Table 2. The system size and a number of cases in each group are shown in the Table 4. Moreover, we investigate the differences among three groups with respect to design structural complexity, use intensity of design patterns and the effect of GoF design pattern instances on the structural complexity relevant design metrics.

Table 6. Results of Friedman's test for Design Metrics

Design Metric	Null Hypothesis (HP0)	Chi-squared	p-value	HP0 Decision
MOA	The distribution of MOA is same across all groups	5.08	0.007	Rejected
MFA	The distribution of MFA is same across all groups	7.13	0.02	Rejected
ANA	The distribution of ANA is same across all groups	2.47	0.029	Rejected
CBO	The distribution of CBO is same across all groups	13.06	0.001	Rejected
RFC	The distribution of RFC is same across all groups	5.76	0.005	Rejected
DCC	The distribution of DCC is same across all groups	9.29	0.009	Rejected

A. Differences among groups in term of design quality

In order to evaluate the differences among groups in term of structural complexity assessors, we perform this analysis in two steps. In the first step, we find the mean descriptive statistical value of each design metric for the Group-1, Group-2 and Group-3 respectively, which are shown in the Table 5. Subsequently, in the second step, we perform a non-parametric Friedman's test in order to investigate the differences between mean values of design metrics across the groups. The null hypothesis (HP0) for each design quality attribute and its decision is shown in the Table 6.

We can conclude from the findings (presented in the Table 5 and Table 6) that there is evidence of differences in the structural complexity of Group-1, Group-2, and Group-3 projects. Therefore, we can also conclude that distribution of design structural complexity is highly related to the confounding effect of system size.

B. Differences among groups in term of use intensity of design patterns

In this analysis, we investigate that difference in the structural complexity distribution (in term of assessors) among Group-1, Group-2, and Group-3 projects can be attributed to use intensity of the GoF design pattern if we find differences in their use intensity. We perform this analysis in two steps. In the first step, we find the mean descriptive statistical value of employed design pattern instances for Group-1, Group-2, and Group-3 respectively, which are shown in the Table 7. Subsequently, in the second step, we perform a non-parametric Friedman's test in order to investigate the differences between mean values employed design pattern instances across the groups. The null hypothesis (HP0) for each design pattern and its decision is shown in the Table 8.

Table 7. Descriptive Statistics of GoF Design Patterns

Design Patterns	Mean Values		
	Group-1	Group-2	Group-3
Singleton	21.75	80.45	195.85
Factory Method	10.24	93.72	239.36
Prototype	10.89	31.62	253.79
Adapter-Command	22.58	23.41	25.36
Observer	5.59	89.56	427.69
State-strategy	19.79	23.85	13.43
Template Method	25.15	72.28	150.03
Visitor	1.12	2.19	9.01
Composite	0.00	26.87	241.72
Decorator	0.43	73.80	231.21
Proxy	5.59	99.79	311.99
Proxy2	0.23	22.66	154.14

Table 8. Results of Friedman's test for GoF Design Patterns

Design Pattern	Null Hypothesis ($HP_{0)}$	Chi-Sq	p-value	HP_0 Decision
Singleton	The use intensity of Singleton is same across all groups	13.06	0.00	Rejected
Factory Method	The use intensity of Factory Method is same across all groups	18.37	0.00	Rejected
Prototype	The use intensity of Prototype is same across all groups	0.89	0.01	Rejected
Adapter	The use intensity of Adapter is same across all groups	12.23	0.03	Rejected
Observer	The use intensity of Observer is same across all groups	18.49	0.00	Rejected
State-strategy	The use intensity of State-strategy is same across all groups	12.24	0.03	Rejected
Template Method	The use intensity of Template Method is same across all groups	14.36	0.00	Rejected
Visitor	The use intensity of Visitor is same across all groups	0.00	0.10	Accepted
Composite	The use intensity of Composite is same across all groups	7.84	0.03	Rejected
Decorator	The use intensity of Decorator is same across all groups	23.05	0.00	Rejected
Proxy	The use intensity of Singleton is same across all groups	15.26	0.00	Rejected
Proxy2	The use intensity of Singleton is same across all groups	1.75	0.42	Accepted

The findings of Table 7 and Table 8 suggest the differences among Group-1, Group-2, and Group-3 in term of use intensity of design patterns, such as in the Table 7, we can observe the incidence of employed design pattern instances is high in Group-3 as compared to Group-1 and Group-2, which can refer the confounding effect of system size.
Subsequently, the hypothesis testing results (shown in the Table 8) of each design pattern suggest us that the structural complexity across the groups can be attributed to use intensity of the design patterns.

C. Differences in the effect of design patterns on Assessors in Groups.

In the section 5.1, we have investigated that design patterns such as Singleton, Factory Method, Adapter-Command, and Template have statistically strong significant correlation with structural complexity assessors (also shown in the Table 3).

Table 9. Spearman's Correlation across the Groups

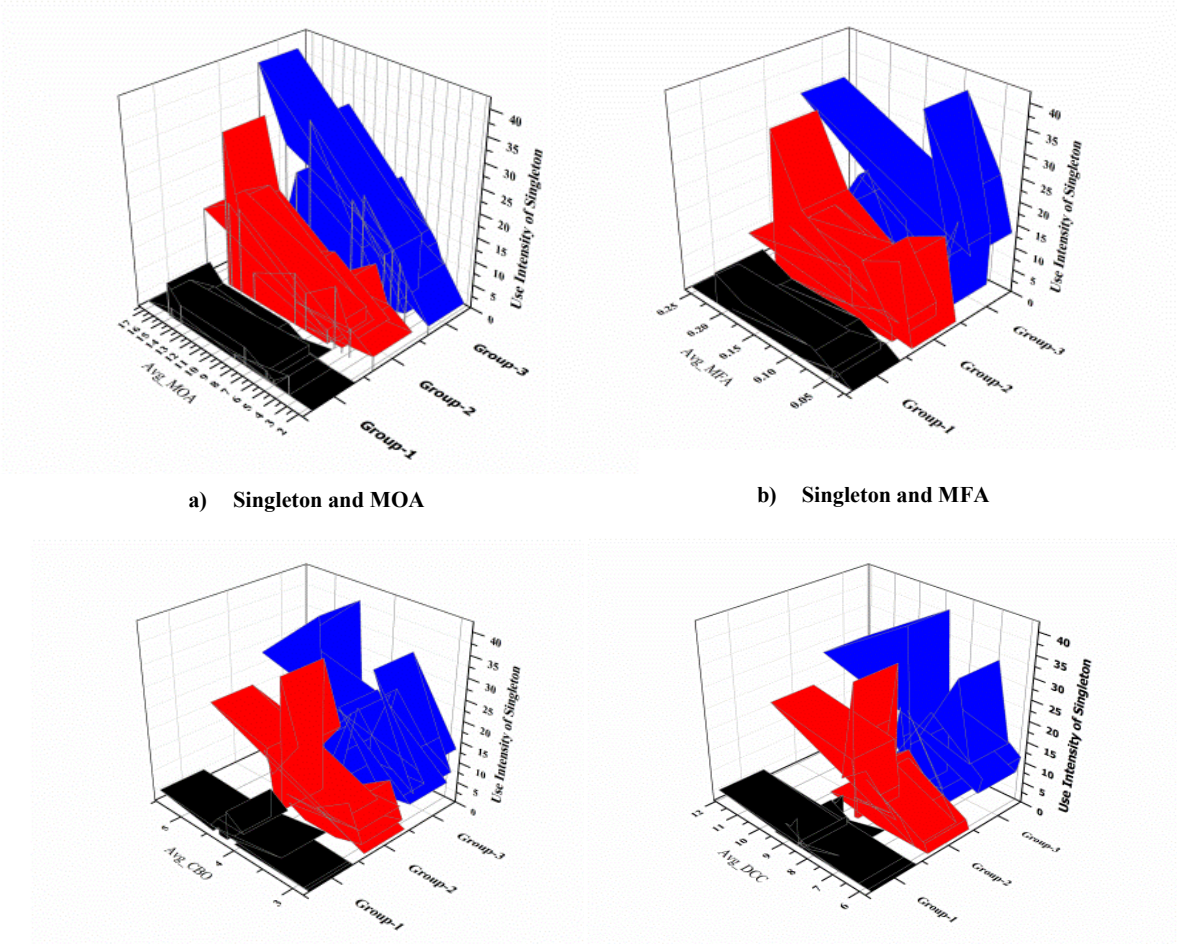
Design Pattern :: Quality Attribute	Group	Corr. Coeff (ρ)	Sig (p)
Singleton :: MOA	Group-1	0.56	0.02
	Group-2	0.43	0.04
	Group-3	0.15	0.22
Singleton :: MFA	Group-1	0.31	0.03
	Group-2	0.23	0.04
	Group-3	0.25	0.08
Singleton :: CBO	Group-1	0.51	0.04
	Group-2	0.18	0.05
	Group-3	0.36	0.06
Adapter-Command :: MOA	Group-1	0.57	0.02
	Group-2	0.23	0.04
	Group-3	0.63	0.01
Adapter-Command :: MFA	Group-1	0.21	0.05
	Group-2	0.32	0.06
	Group-3	0.31	0.07
Adapter-Command :: CBO	Group-1	0.36	0.03
	Group-2	0.43	0.04
	Group-3	0.49	0.05
Factory Method :: MOA	Group-1	0.59	0.01
	Group-2	0.31	0.03
	Group-3	0.32	0.03
Factory Method:: MFA	Group-1	0.32	0.07
	Group-2	0.34	0.06
	Group-3	0.38	0.06
Factory Method :: CBO	Group-1	0.44	0.03
	Group-2	0.78	0.00
	Group-3	0.47	0.05
Template :: MOA	Group-1	0.46	0.03
	Group-2	0.37	0.06
	Group-3	0.14	0.12
Template :: MFA	Group-1	0.46	0.06
	Group-2	0.24	0.08
	Group-3	0.13	0.11
Template :: CBO	Group-1	0.33	0.03
	Group-2	0.13	0.09
	Group-3	0.18	0.11

Subsequently, in this analysis, we further investigate that how system size influences the significant strength of correlation between the number of employed instances of these design patterns (such as Singleton, Factory Method, Adapter-Command and Template) and assessors. Again, we perform non-parametric (distribution free) Spearman's correlation to measure the strength of the relationship between use intensity of design patterns and assessors for each group. The detailed results of this analysis are shown in the Table 9. We have not included the correlation result of design patterns (Singleton, Adapter-Command, Factory Method, and Template) with design metric DCC due to Table 9 adjustment on the page.
From these results, we can conclude that in the case of group-1 projects, employed instances of Singleton, Adapter-Command, State-Strategy and Template design patterns have a significant effect on the structural complexity assessors, while in the case of group-2 and group-3 projects, we can observe this effect at different significant levels. For example, the effect of employed instances of Singleton on the assessor CBO can be observed as ρ=0. 43 (group-2) and ρ=0.49 (group-3) at significant level p=0.04 and p=0.01 respectively. Subsequently, we also explore interesting findings to differentiate the effect of the number of employed design pattern instances on the assessors with respect to system size through the 3-D area chart plot. The y-axis of each plot presents the use intensity (with a fraction of a number of classes and employed instances) of a design pattern; x-axis presents the mean values of a assessor, and overall plot area depicts the effects across the groups. The value of assessor (x-axis) can be interpreted with respect to the corresponding use intensity of a design pattern (y-axis). For example, drop lines on XY faces of Figure 1(a-d) present the influence of use intensity of Singleton on the CBO. Subsequently, we can also observe an approximate consistent increase in assessors with frequent use of the Singleton design pattern in group-1 as compared to group-2 and group-3.
Finally, from the results shown in the Table 5-9, it can be concluded that system size has more concern in order to investigate the effect of use intensity of the GoF design patterns on the structural complexity assessors at the system level.

5.3 Respond to RQ-3.

In agile development, there are two common practical approaches to address decline quality of software systems with the passage of time such as refactoring and employ design patterns. In this analysis, we investigate the influence of the change in a number of employed design pattern instances on the structural complexity in the subsequent releases of a system. In this analysis, we consider nine subsequent releases of an open source project

named velocity, and create a dataset with respect to the structure defined in the Table 2. Subsequently, during analysis, we observe no change in the number of employed instances of the GoF design patterns except Adapter-Command, State-Strategy and Template design patterns. We also observe the significant effect of use intensity of Adapter-Command, State-Strategy, and template design patterns on the structural complexity assessors through Spearman correlation. For example, the use intensity of Adapter-Command has a significant correlation with design metrics MOA (ρ=0. 63), MFA (ρ =0.51), CBO (ρ=0.66), DCC (ρ=-0.52), ANA (ρ=0. 46) and RFC (ρ=-0.77) at level $p<0.05$. Moreover, we plot a 2-D line chart in order to present the change in structural complexity assessor with respect to changes in the employed design pattern instances in subsequent releases. For example, in the Figure 2, we present the monotonic relationship between use intensity of Adapter-command pattern and the variations in the assessors. The x-axis of plots (Figure 2) presents the use intensity value of design patterns (from a low value to high) and y-axis present the assessor values. From the Figure 2, a consistent increase in the structural complexity (relevant to MOA, CBO, and DCC assessors) can be observed in the subsequent release of velocity system, and the same case is not true for MFA. From Spearman correlation results and plots shown in the Figure 3, it can be suggested that the change in the employed instances of design patterns in a subsequent release of an application also affect its structural complexity, such as improvement in functional abstraction of system design components due to frequent use of Adapter-Command design pattern can be observed in the Figure 2-c.



a) **Singleton and MOA**

b) **Singleton and MFA**

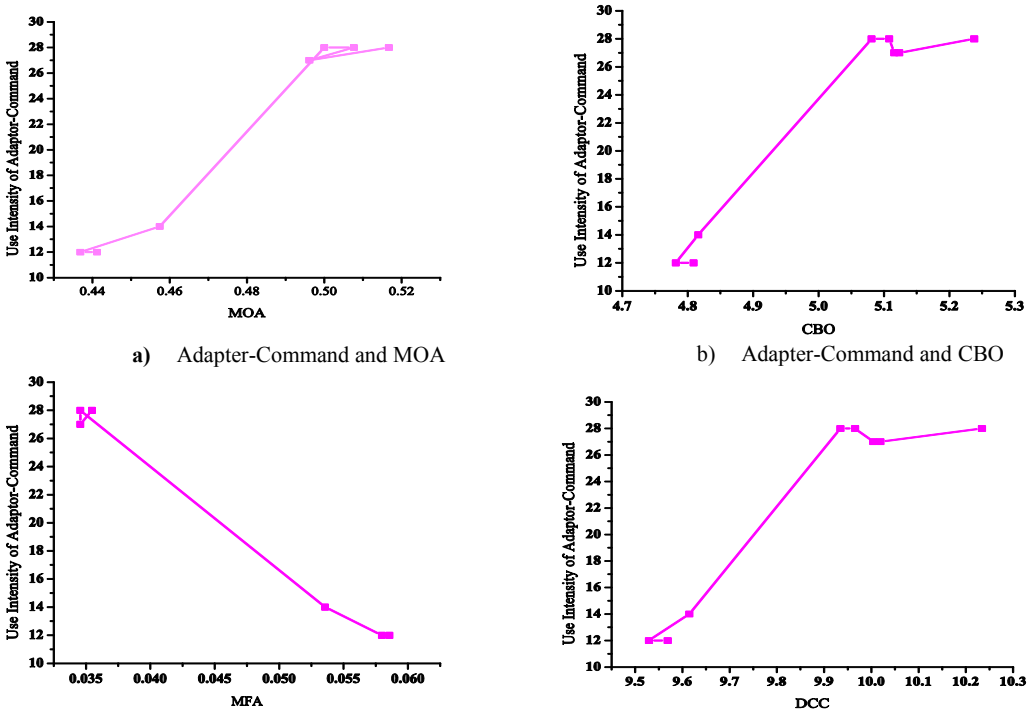
c) **Singleton and CBO**

c) **Singleton and DCC**

Fig. 1. 3-D Chart Areas

6. THREAT TO VALIDITY

In this section, we describe some internal and external threats to validity in order to present our case study. The first threat is related to the size of the dataset for empirical investigation of the effect of use intensity of design patterns on the structural complexity. We select 51 open source projects which are further classified into groups with respect to system size. The results can alter by increasing the number of cases in the proposed case study. The second threat is about the selected design patterns, our results on 12 design patterns cannot be generalized to the rest of 23 GoF design patterns. The third threat is that the results and conclusion of our research strongly depend on the six structural complexity assessors (i.e. Design metrics) which help to explain the structural complexity of design components (i.e. Classes) of a system, however, an inclusion of other design metrics in the study can aid to alter the existing significant results. The forth threat is related to use of the aggregation method (i.e. Aggregation by simple averaging), which might produce unwanted values in terms of their standard deviation.



a) Adapter-Command and MOA

b) Adapter-Command and CBO

c) Adapter-Command and MFA

d) Adapter-Command and DCC

Fig. 2. Correlation between Use Intensity & Assessors in Subsequent releases of Velocity

7. CONCLUSION

In agile software development, developer seeks opportunities to apply design patterns in a more systematic way to investigate the structural complexity and assure the design quality at the system level. In this regard, the information about the effect of employed instances of design pattern on structural complexity assessors at the system level can help the developers to monitor the systems and relevant activities in real life context. In this paper, we present a holistic multiple-case study with three types of analysis, which can aid developers to evaluate the effectiveness of employed instances of the Gang-of-Four (GoF) design patterns on the structural complexity associated with design properties (such as abstraction, coupling, inheritance etc.) in the domain of open source applications. The cases and unit of analysis of the case study are 51 open source projects and nine subsequent releases of an open source project named Velocity. The result of this study suggests that structural complexity associated with aggregation, coupling, functional abstraction design properties has a significant relationship with the employed instances of Template, Adapter-Command, and Singleton and Factory Method

design patterns; however, it is affected by the confounding effect of system size. In the case of third analysis with the subsequent releases of an open source project named velocity, we observed the use intensity of Singleton, Adapter-Command, and Factory Method design patterns can increase the structural complexity in term of coupling and aggregation as compared to functional abstraction. Subsequently, in the future, we will investigate the effect of the GoF design pattern on the external design quality and its relation with structural complexity at class level rather than system level.

ACKNOWLEDGEMENT

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 11208017 and 11214116), and the research funds of City University of Hong Kong (No. 7004683).

8. REFERENCES

[1] A. Ampatzoglou, S. Charalampido, I. Stamelos, Research State of the art on GoF Design Patterns: A mapping study, Journal of System and Software, 86, p. 1945-1964, 2013.

[2] C. Zhang, and D. Budgen, A survey of experienced user perceptions about software design patterns. Information and Software Technology, 55 (5): p. 822-835, 2013.

[3] S.R Chidamber, C.F. Kemerer, A Metrics Suite for Object-Oriented Design, IEEE Transaction on Software Engineering, 20, p. 476–493, 1994.

[4] R. Subramanyam and M.S. Kishnan, Empirical Analysis of CK Metrics for Object-oriented Design Complexity : Implication for Software Defects, IEEE Transaction on Software En-gineering, 2003, p. 297-310, 2003.

[5] J. Bansiya, and C. G. Davis, A Hierarchical Model for Object-Oriented Design Quality As-sessment. IEEE Transaction on Software Engineering, 28(1): p. 4-17, 2002.

[6] E. Gamma, RF. Helm, R. Johnson, J. Vlissides, Design Patterns : Elements of Reusable Object-Oriented Software, Boston: Addison-Wesley. 1995.

[7] A. Ampatzoglou, A. Chatzigeorgiou, Evaluation of object-oriented design patterns in game development, Information and Software Technology,49 (5), p. 445-454, 2007.

[8] M. Vok, W. Tichy, D. I. K. Sjoberg, M. Aldrin, A Controlled Experiment Comparing the Maintainability of Programs Designed with and without Design Patterns; A Replication in a Real Programming Environment. Empirical Software Engineering, 9 (3): p. 149-195, 2004.

[9] L. Prechelt,B. Unger, W.F. Tichy, P. Brossier, L. G. Votta, A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solutions. IEEE Transactions on Software Engineering, 27(12): p. 1134-1144, 2001.

[10] S. Jeanmart, Y-G. Guehenuec, H. Sahraoui, N. Habra. Impact of the visitor pattern on program comprehension and mainte-nance, Proceedings of 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009.

[11] P. Runeson, M. Host, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, John Wiley and Sons, 2012.

[12] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, S. R. Halkids, Design Pattern Detection using Similarity Scoring, IEEE Transaction on Software Engineering, 32(11), pp. 896-909, 2006.

[13] R. Terra, L. F. Miranda, M. R. Valente, R. S. Bigonha, Qualitas.class Corpus: A compiled version of the Qualitas Corpus," Software Engineering, Engineering Notes, 38(5), pp. 1–4, 2013.

[14] P. Sfetsos, A. Ampathoglou, A Chatzigeorgiou, I. Deligiannis, I, Stamelos, A Comparative Study on the Effectiveness in Software Libearies and Standalone Applications, 9th International Conference on the Quality of Information and Communication Technology, 2014.

[15] S. Hussain, J. Keung, A. A. Khan, The Effect of Gang-of-Four Design Patterns Usage on Design Quality Attributes, Proceeding of QRS-2017, 27-30 Jul 2017.

[16] C. Weber, What Is Complexity?, In Proceedings of the 15th International Conference on Engineering Design (ICED 05), Institution of Engineers, Melbourne, 2005.

[17] U. Lindemann, M. Maurer, T. Braun, Structural Complexity Management - An Approach for the Field of Product Design, Springer, 2008.

[18] S. A. Sheard and A. Mostashari, A Complexity Typology for Systems Engineering, Systems Engineering, 2010.