Exploring the Relationships between Design Metrics and Package Understandability: A Case Study

Conference Paper · June 2010

DOI: 10.1109/ICPC.2010.43 · Source: DBLP

Mahmoud Elish
Gulf University for Science and Technology (Kuwait)

18th IEEE International Conference on Program Comprehension

Exploring the Relationships between Design Metrics and Package Understandability: A Case Study

Mahmoud O. Elish
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
elish@kfupm.edu.sa

Abstract — **In object-oriented designs, packages represent important high-level organization units that group classes. This paper explores the relationships between five package-level metrics and the average effort required to understand a package in object-oriented design. These metrics measure different structural properties of a package such as size, coupling and stability. A case study was conducted using eighteen packages taken from two open source software systems. Correlation, collinearity, and multivariate regression analyses were performed. The results obtained from this study indicate statistically significant correlation between most of the metrics and understandability of a package.**

Keywords - software metrics; software comprehension; object-oriented software; packages.

I. INTRODUCTION

As object-oriented software applications grow in size and complexity, they require some kind of high-level organization. Classes, while a very convenient unit for organizing small applications, are too finely grained to be used as the sole organizational unit for large applications. This is where packages come in; packages are mechanism for organizing classes into namespaces. A package is a basic development unit that can be separately created, maintained, released, tested, and assigned to a team [3]. A Java package is defined as follow [7]: "A java package is a group of classes that are related by purpose or by application. Classes in the same package have special access privilege with respect to one another and may be designed to work together closely."

Studying and analyzing packages in object-oriented software in order to evaluate the quality of software is becoming increasingly important as object-oriented paradigm continues to increase in popularity, the size and number of packages of these software increases [4, 6, 8]. Consequently, several package-level metrics have been proposed and used to characterize the attributes of packages in object-oriented software design. Many of these attributes have relation, in one way or the other, with the quality of the software being produced. However, some of these metrics may or may not really measure the intended quality attributes of software. Thus, empirical validation is necessary to demonstrate the usefulness of these metrics in practical applications [1], i.e. explore the relationships between these metrics and some important external quality attributes. This paper explores the relationships between a suite of five metrics, proposed by Martin [6], and package understandability.

The rest of this paper is organized as follows. Section II reviews related work. Section III defines the package-level metrics under investigation. Section IV describes the conducted case study with discussion and analysis of the results. Section V concludes the paper.

II. RELATED WORK

Package-level metrics such as coupling and stability metrics have been studied by many researchers. Martin [6] introduced a set of package-level metrics that are related to dependencies, abstraction and stability. However, these metrics have not been empirically validated.

Zimmermann et al. [10] aggregated some method and class-level complexity metrics at package-level and correlated them with pre- and post-release faults of a package. The results showed that the combination of complexity metrics can predict faults. However, the predictions were far from being perfect [10].

D'Ambros and Lanza [2] proposed evolution radar to understand the package coupling. Ducasse et al. [4] proposed a generic visualization technique that can be used to visualize, analyze and understand package relationships. Wilhelm and Diehl [9] used Martin's [6] and size metrics to build a tool that helps to control package dependencies.

None of the previous studies have explored the relationships between design metrics and package understandability; except Gupta and Chhabra [5] who proposed a package coupling metric and empirically validated it against package understandability. Their study was limited to one metric, and they performed only correlation analysis. This case study, however, considers five metrics and performs correlation, collinearity, and multivariate regression analyses.

III. PACKAGE-LEVEL METRICS

In this study, a suite of five popular and well-known package-level metrics, proposed by Martin [6], are explored as indicators of the effort required to understand a package. These metrics measure different structural properties of a package such as size, coupling and stability. They are defined next.

978-0-7695-4113-6/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPC.2010.43

A. *Number of Classes (NC)*

The *NC* metric for a package is defined as the number of concrete and abstract classes (and interfaces) in the package. It is a measure of package size.

B. *Afferent Couplings (Ca)*

The *Ca* metric for a package is defined as the number of other packages that depend upon classes within the package. It measures the incoming dependencies (fan-in).

C. *Efferent Couplings (Ce)*

The *Ce* metric for a package is defined as the number of other packages that the classes in the package depend upon. It measures the outgoing dependencies (fan-out).

D. *Instability (I)*

The *I* metric for a package is defined as the ratio of efferent coupling (*Ce*) to total coupling (*Ce* + *Ca*) for the package; such that $I = Ce / (Ce + Ca)$. This metric is an indicator of the package's resilience to change. The range for this metric is zero to one, with zero indicating a completely stable package and one indicating a completely unstable package.

E. *Distance (D)*

The *D* metric for a package is defined as the perpendicular distance of the package from the idealized line ($A + I = 1$), where *A* is the percentage of the abstract classes to the total number of classes in the package. This metric is an indicator of the package's balance between abstractness and stability. A package squarely on the main sequence is optimally balanced with respect to its abstractness and stability. The range for this metric is zero to one, with zero indicating a package that is coincident with the main sequence and one indicating a package that is as far from the main sequence as possible.

IV. The Case Study

This case study aims to explore the relationships between the five package-level metrics, described in the previous section, and package understandability. For this purpose, two open source software systems were chosen: XGen Source Code Generator[1] and Jakarta Element Construction Set (ECS)[2]. These two systems were chosen because of the availability of data about the average effort required to understand their packages, which was published recently by Gupta and Chhabra [5]. XGen has 6 packages with 52 classes, and Jakarta ECS has 12 packages with 418 classes.

A. *Independent and Dependent Variables*

The independent variables are the five package-level metrics under investigation, i.e. *NC*, *Ca*, *Ce*, *I*, and *D* metrics. Table 1 provides descriptive statistics of these metrics that were collected for the 18 packages of XGen and Jakarta ECS. The number of classes in these packages varies from 1 to 91 with an average of around 23 classes. The average values of the *Ca* and *Ce* metrics indicate that these packages have more efferent coupling than afferent coupling. Accordingly, the average value of the *I* metric is above 0.5.

Table 1. Descriptive statistics

Metric	Min	Max	Avg.	StDev
NC	1.0	91.0	22.6	29.9
Ca	0.0	12.0	2.2	3.3
Ce	2.0	22.0	5.3	4.5
I	0.3	1.0	0.8	0.2
D	0.0	0.7	0.2	0.2

The dependent variable is the average effort required to understand a package. In a recent study by Gupta and Chhabra [5], they measure the average effort required to understand the 18 packages of XGen and Jakarta ECS as follows. They assigned these 18 packages to three teams and each team having three members, where members of all the teams have almost similar experience of Java programming. All the teams are required to fully understand the functionality of these packages and assess the effort required to understand each package and rank the effort from 1 to 10. A higher rank indicates that more effort spent on understanding the package. Table 2 provides the ranked efforts to understand the 18 packages by the three teams and the average, as published in [5]. It can be observed that the average effort to understand these packages varies from 1.0 to 9.0 with a mean of 3.6. Also, the difference between the rankings by the three teams for each package is at most two.

Table 2. Ranks assigned for understandability of packages

No.	Package Name	Team			Average Effort
		1	2	3	
1	workzen.xgen.ant	3	1	2	2.0
2	workzen.xgen.engine	2	2	3	2.3
3	workzen.xgen.loader	5	5	6	5.3
4	workzen.xgen.model	6	5	7	6.0
5	workzen.xgen.test	5	4	4	4.3
6	workzen.xgen.util	2	3	3	2.7
7	org.apache.ecs.examples	2	1	1	1.3
8	org.apache.ecs.factory	1	1	1	1.0
9	org.apache.ecs.filter	3	2	1	2.0
10	org.apache.ecs.html	8	9	10	9.0
11	org.apache.ecs.html2ecs	1	1	2	1.3
12	org.apache.ecs.jsp	2	1	2	1.7
13	org.apache.ecs.rtf	2	2	3	2.3
14	org.apache.ecs.storage	2	1	3	2.0
15	org.apache.ecs.vxml	1	1	2	1.3
16	org.apache.ecs.wml	7	8	9	8.0
17	org.apache.ecs.xhtml	8	9	10	9.0
18	org.apache.ecs.xml	4	3	5	4.0

B. *Hypotheses*

This case study tested the following null hypotheses:

- *Hypothesis 1*: There is no correlation between *NC* and the effort required to understand a package.

[1] http://sourceforge.net/projects/xgen/
[2] http://jakarta.apache.org/ecs/index.html

- *Hypothesis 2*: There is no correlation between *Ca* and the effort required to understand a package.
- *Hypothesis 3*: There is no correlation between *Ce* and the effort required to understand a package.
- *Hypothesis 4*: There is no correlation between *I* and the effort required to understand a package.
- *Hypothesis 5*: There is no correlation between *D* and the effort required to understand a package.

C. Correlation Analysis

The correlation analysis aims to determine if each individual package-level metric is significantly related to the average effort required to understand the package. This analysis was performed to test the hypotheses. For this purpose, Spearman's rank correlation was performed due to the nonparametric nature of the metrics. The significance of the correlation was tested at 95% confidence level (i.e. p-level ≤ 0.05). The results obtained by applying this test are given in Table 3, where bold values indicate statistically significant correlations. All metrics, except the *Ce* metric, were found to be significantly correlated with the average effort required to understand a package. Thus, the null hypothesis 3 is accepted, but the null hypotheses 1, 2, 4, and 5 are rejected.

Table 3. Spearman correlation test results

Metric	Correlation Coefficient (r)	p-level
NC	**0.625**	**0.005548**
Ca	**0.599**	**0.008660**
Ce	-0.318	0.198076
I	**-0.617**	**0.006351**
D	**0.638**	**0.004372**

The positive significant correlation between the *NC* metric and package understandability suggests that the bigger the size of a package in terms of the number of classes the more effort is required to understand it. This is not a surprising result.

The positive significant correlation between the *Ca* metric and package understandability suggests that the higher the incoming dependencies of a package the more effort is required to understand it. This observation can be explained; high *Ca* of a package indicates that it is a service package and therefore requires more effort to understand its services.

The negative correlation between the *Ce* metric and the effort required to understand a package was surprising. We were expecting that as a package depends upon more packages, it requires more effort to understand it. The result can be however explained; high *Ce* of a package indicates high reusability of services provided by other packages. These services, if well-understood, will decrease the time required to understand the reusing package(s).

The negative significant correlation between the *I* metric and the effort required to understand a package seems surprising. However, stability here means more difficult to change. The negative correlation can thus be interpreted as follows; the easier to change a package the less effort is required to understand it.

The positive significant correlation between the *D* metric and package understandability suggests that the more the package's unbalance between abstractness and stability the more effort is required to understand it.

D. Collinearity Analysis

Table 4 shows Spearman correlations between all the metrics, where bold values indicate statistically significant correlations at 95% confidence level (i.e. p-level ≤ 0.05). It can be observed that there are significant correlations between *I* and *Ca*; *D* and *Ca*; and *I* and *D* metrics. These high correlations might cause collinearity. Accordingly, the condition number and the Variance Inflation Factor (VIF) values for the metrics were calculated. Table 5 shows the condition number and VIF analysis results. The first VIF column shows the VIF analysis and the condition number for all the metrics, while the second VIF column shows the result after the *I* metric was dropped since it has the highest VIF value in the first column.

The condition number was less than 30 (a threshold suggested in the literature) after dropping the *I* metric. The results of the collinearity analysis therefore suggest not using the *I* metric in conjunction with any of the other four metrics (*NC*, *Ca*, *Ce*, and *D*) in prediction models. The *I* metric was consequently dropped from the subsequent analyses. The set of the other four metrics does not suffer from collinearity.

Table 4. Spearman correlation among the metrics

	NC	Ca	Ce	I	D
NC	1.000				
Ca	0.136	1.000			
Ce	-0.294	0.157	1.000		
I	-0.102	**-0.939**	0.086	1.000	
D	0.156	**0.924**	-0.086	**-0.984**	1.000

Table 5. Condition number and VIF analysis results

Metric	VIF	VIF (w/o I)
NC	1.079	1.045
Ca	1.964	1.963
Ce	1.265	1.263
I	4113.451	N/A
D	4110.298	1.826
Condition number	*601.247*	*4.801*

E. Multivariate Regression Analysis

The multivariate analysis was performed to construct different multivariate linear regression models for predicting the effort required to understand a package as a function of the independent variables. Since there are four independent variables (*NC*, *Ca*, *Ce*, and *D*), after excluding the *I* metric, 15 different prediction models were built that represent all possible combinations of these variables. A leave-one-out cross-validation procedure was used to evaluate and compare the accuracy of these prediction models.

The accuracy of the prediction models were evaluated based on de facto standard and commonly used measures: Mean Magnitude of Relative Error (MMRE) and prediction at level 0.25 (Pred(0.25)) measures.

Table 6 shows the MMRE and Pred(0.25) values for each of the 15 prediction models, where 'x' indicates the presence of the independent variable in the corresponding model. It can be noticed that Model 5 achieved the best MMRE value (0.464), followed by Models 11, 12, and 13 which achieved very competitive MMRE values. The best model in terms of Pred(0.25) is Model 12, which achieved a value of 55.6%.

It can be observed from Table 6 that those models that include the *NC* metric outperformed those that do not include it as independent variables. However, considering only the package size as input to prediction models for package understandability is not enough. Improved prediction could be achieved by considering other structural properties in addition to size. For instance, improved prediction was achieved by Models 5, 7, 11, 12, 13 and 15 compared to Model 1.

Table 6. Evaluation of the prediction models

Model	**Independent Variables**				**Prediction Evaluation Measures**	
	NC	**Ca**	**Ce**	**D**	**MMRE**	**Pred(0.25)**
1	x				0.678	27.8%
2		x			0.947	11.1%
3			x		0.939	11.1%
4				x	0.950	22.2%
5	x	x			0.464	44.4%
6	x		x		0.678	27.8%
7	x			x	0.487	44.4%
8		x	x		0.904	5.6%
9		x		x	0.792	27.8%
10			x	x	0.893	16.7%
11	x	x	x		0.467	50.0%
12	x	x		x	0.467	55.6%
13	x		x	x	0.468	50.0%
14		x	x	x	0.943	11.1%
15	x	x	x	x	0.508	44.4%

F. *Threats to Validity*

This study has a number of limitations that are not unique to it but are common with most of the case studies in the literature. We did not measure package understandability in this study. Instead, we used recently published dataset [5]. Given that the difference between the rankings by the three teams for each package is at most two; we believe that the data is reliable.

This study was limited to five package-level metrics. These metrics are popular and measure different structural properties of a package. We plan to investigate more metrics as a future work. This study was also limited to a small-size dataset that consists of eighteen packages due to the availability of data about the average effort required to understand these packages. However, appropriate statistical analyses were performed in this study.

V. CONCLUSION

This paper has explored the relationships between five package-level metrics and the average effort required to understand a package in object-oriented design. These metrics measure different structural properties of a package such as size, coupling and stability. A case study was conducted using eighteen packages taken from two open source software systems. Correlation, collinearity, and multivariate regression analyses were performed. The results obtained from this study indicate statistically significant correlation between most of the metrics and understandability of a package. In addition, it was observed that considering only the package size as input to prediction models for package understandability is not enough. Improved prediction could be achieved by considering other structural properties in addition to size.

Future works include exploring more package-level metrics; investigating more systems with more packages; exploring the relationships with other software quality attributes; and building computational intelligence models to improve the prediction accuracy.

ACKNOWLEDGEMENT

The author would like to acknowledge the support of King Fahd University of Petroleum and Minerals in the development of this work.

REFERENCES

[1] V. Basili, L. Briand, and W. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering,* vol. 22, pp. 751 - 761, 1996.

[2] M. D'Ambros and M. Lanza, "Reverse Engineering with Logical Coupling," in *13th Working Conference on Reverse Engineering*, pp. 189-198, 2006.

[3] D. D'Souza and A. Wills, *Objects, Components and Frameworks with UML: The Catalysis Approach*: Addison-Wesley, 1998.

[4] S. Ducasse, M. Lanza, and L. Ponisio, "Butterflies: A Visual Approach to Characterize Packages," in *11th IEEE International Software Metrics Symposium*, 2005.

[5] V. Gupta and J. Chhabra, "Package Coupling Measurement in Object-Oriented Software," *Journal of Computer Science and Technology,* vol. 24, pp. 273-283, 2009.

[6] R. Martin, *Agile Software Development: Principles, Patterns and Practices*: Prentice Hall, 2003.

[7] P. Niemeyer and J. Knudsen, *Learning Java*: O'Reilly & Associates, 2005.

[8] R. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed.: Mc Graw Hill, 2005.

[9] M. Wilhelm and S. Diehl, "DependencyViewer - A Tool for Visualizing Package Design Quality Metrics," in *3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis* 2005.

[10] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting Defects for Eclipse," in *3rd International Workshop on Predictor Models in Software Engineering*, 2007.