International Journal of Software Engineering and Its Applications
Vol.8, No.4 (2014), pp.75-84
http://dx.doi.org/10.14257/ijseia.2014.8.4.09

Assessing Package Reusability in Object-Oriented Design

Vinay Singh[1] and Vandana Bhattacherjee[2]

[1]*Usha Martin Academy, Ranchi, India*
[2]*Birla Institute of Technology, Ranchi, India*
[1]*mailtovsingh@yahoo.co.in*, [2]*bhattacherjeev@yahoo.in*

Abstract

This paper computes reusability at the package level. To obtain the package reusability, authors has considered four design size package metrics. These metrics are then mapped to the Object-Oriented design properties: cohesion, coupling, messaging and the design size. The package reusability is empirically validated through three successive versions of the JFree Chart.

Keywords: *Coupling,Cohesion,Messaging, Reusability,Software metrics*

1. Introduction

Object-Oriented measurement can be done mostly at two levels: at fine grains and at the coarse grain levels. Fine grains measurement is at class level, which, is further at method level and attribute level. A rich set of research has been done at the class level. Coarse grains measurement is at Package and System level. The package is a collection of classes and interfaces related by purpose or by application. Classes in package work together closely and provide specific functionality to the system. Packaging in Object-Oriented design plays an important role as it is the container of the functionalities (classes) of the software systems. Measuring quality at package level is always a challenging task for the researchers and practitioners. Reusability is the process of creating a new product from an existing one without significant effort. The new product can contain the feature of the old one, but should also have new features in it. The software is said to be reusable if it is easy to understand and is less complexed. It is always important to keep track of the extendibility and reusability of the packages in the successive releases of a software system.

The measurement at package level was first proposed by [3]. He has defines four principles:

- Stable-abstraction principle: It states that the stable package should also be abstract, so that its stability does not preclude it from being carried. He defines the abstraction by dividing the summation of afferent coupling and efferent coupling by efferent coupling with the range [0, 1]. 0 indicates maximally stable package.

- Dependency inversion principle: It states that the packages should depend on abstract entities, not concrete ones.

- Acyclic dependency principle: It states that the packages must form no cycle's means that all packages in the system should be self-contained as possible.

- Encapsulation principle: It states that each of the modules encapsulates implementation details that are not visible to the user of the module.

ISSN: 1738-9984 IJSEIA

All the four metric principles focused on low coupling and high cohesion among the packages.

The maximum possible intra-connectivity dependencies as n^2 where n is the number of classes in the package [4]. The component cohesion metric proposed by [5] measures the cohesion of package by the number of possible coupling relationships among the classes is n (n-1) where n is the number of classes. Martin [3] measures cohesion of a package as the average number of internal relationships per class in the package. Khan in his MS Thesis [6] proposed design level package metrics. He defined the internal package coupling metric that counts the total coupling between the classes of the same package. The package cohesion metric is defined in [7] as a proportional to the number of internal dependencies within the package. They assume one interaction to a class outside the package. The survey of the existing package cohesion metric and the evaluation of attribute based framework has been studied in [8].

Package coupling metrics can be measured by the incoming and outgoing dependencies termed as afferent and efferent coupling [3]. Afferent coupling (CA) is calculated by the number of incoming dependencies; on the other hand, Efferent coupling (Ce) is calculated by the number of outgoing dependencies. Efferent coupling can be reduced by refactoring of classes inside a package into smaller classes. High values of Ca are acceptable when the package is relatively abstract than concrete.

An improved hierarchical model QMOOD for the assessment of quality attributes reusability, flexibility understandability, functionality, extendibility and effectiveness in Object-Oriented designs has been validated at class level [1]. The QMOOD model has been adopted which has been carried out at four levels (Identifying design quality attributes, identifying Object-Oriented design properties, identifying Object-Oriented design metric and identifying Object-Oriented design components). A suite of class level metrics at the package level to assessing software quality has been studied in [2], the validation of these metrics has been done with three successive versions of an open source projects JFreeChart.

In this paper an attempt is to find the package reusability. An empirical study has been carried out with 16 packages from 3 successive versions of JFreeChart. The rest of the paper is organized as follows: Section 2 present the package metrics, Section 3 identifies Object-Oriented methodologies whereas Section 4 carried the empirical study, paper has been concluded in Section 5.

2. Package Metrics

2.1. Existing Package Metric

The existing design package metrics are presented in Table 1.

Table 1. Existing Package Metrics Description

METRIC	NAME	DESCRIPTION	Reference
CE	Efferent Coupling	This metric is a count of the number of packages that this package depends upon.	[3]
$LCOM_{avg}$	Lack of cohesion	This metric is a count the average number of Lack of Cohesion of methods of classes ($LCOM_{avg}$) in package P is defined $\sum_{i=1}^{n}\left(\left(\frac{1}{INST}\right) * (Numref - NOMT)/(1 - NOMT)\right)$ Where, Numref is the sum of the number of attribute references in each of the methods in the class INST is the number of instance variables defined in the class. NOMT is the number of methods in the class.	[2]

2.2. Proposed Package Metric

Design size package (DSP) and Package interface size (PIS) are proposed for measuring reusability.

Design Size Package (DSP)

This metric is a count of the number of classes and interfaces in package p. The metric values are calculated as:

$DSP = \left[\sum_{i=1}^{n} NC_i + IP_i\right] \in P$ Where, NC_i is the number of classes i in package p and IP_i is the number of interfaces i in package p.

Package Interface Size (PIS)

This metric is a count of the number of public classes in package p. The metric values are calculated as:

$PIS = \left[\sum_{i=1}^{n} PNC_i\right] \in P$ Where, PNC_i is the number of public classes i in package p.

3. Object-Oriented Methodology

The definition of key Object-Oriented properties that influences the reusability given as follows:

- Design size – A measure of the number of classes and interfaces used in a package design.
- Coupling – Package X is coupled to Package Y if and only if X sends a message to Y.
- Cohesion – The degree to which the methods of classes within a packages are related to one another.

- Messaging- A count of the number of public classes that are available as services to other packages. This is a measure of the services that a package provides.

Table 2. Design Metrics for Design Properties

Design Property	Derived Design Metrics
Design Size	DSP (Number of Classes and Interfaces)
Coupling	CE (Efferent Coupling)
Cohesion	$LCOM_{avg}$ (Package Cohesion
Messaging	PIS (Package Interface Size)

The mapping of Object-Oriented design properties to package design metrics are presented in Table 2.

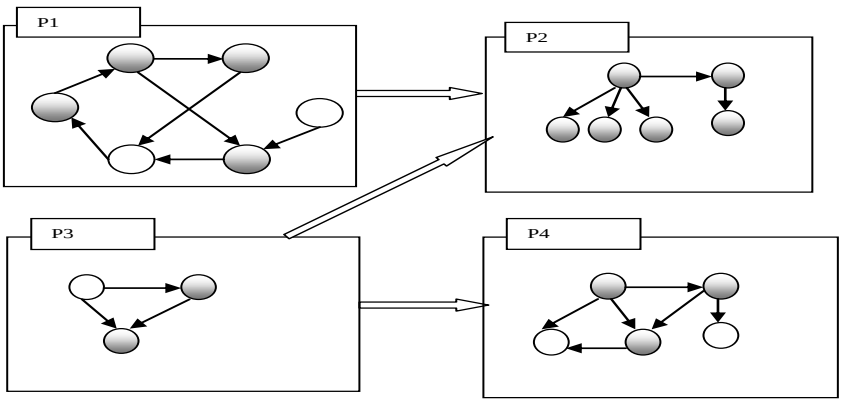
The four design properties, i.e. design size, cohesion, coupling and messaging influence package reusability [1]. Low coupling and high cohesion are considered good for reusability and understandability. Intra package communication by message passing is possible when access control modifiers of classes are public and therefore, messaging directly influences the functionality and effectiveness and helps promote reusability.

The Reusability can be measured by the following index computation equation [1].

$$Reusability = -0.25 * Coupling + 0.25 * Cohesion + 0.5 * Messaging + 0.5 * Design\ Size \quad (i)$$

The weighted values of +1 or +0.5 indicate positive influence and -1 or -0.5 indicate the negative influence.

Illustration



- **Circle indicate classes and interfaces**
- **Solid circle indicate public class**
- **Solid arrow denotes interaction among classes**
- **Hollow arrows denotes outside package dependencies**

Figure 1. System X with Different Amount of Interaction among Packages

Figure 1 shows the system X, with different amount of interaction among packages. The design package metric value of system X is calculated as follows: The total number of classes

and interfaces in package P1 is 6 hence the value of DSP=6. Similarly for package P2 DSP=6, for package P3 DSP=3 and for package P4 DSP=5. The number of public classes in package P1 is 4 hence the PIS=4. Similarly for package p2 PIS=6, for package P3 PIS=2 and package P4 PIS=3. The number of outgoing dependencies (Efferent coupling) for package P1=1 so, Ce=1. Similarly Ce =0 for package P2 because there is no outgoing dependency. The number of outgoing dependencies for package p3 is 2 (P3->P4, P3->P2) hence Ce=2. There is no outgoing dependency for package P4 hence Ce=0. It can be assumed that the $LCOM_{avg}$ for each package is 1. The metric values and the computed reusability values using equation *(i)* for each package of System X are given in Table 3.

Table 3. Reusability Measures

Package	Design Size Package Metrics				
	DSP	**Ce**	**$LCOM_{avg}$**	**PIS**	**Reusability**
P1	6	1	1	4	5.0
P2	6	0	1	6	6.25
P3	3	2	1	2	2.25
P4	5	0	1	3	4.25

Among all the packages in Table 3, P2 has the highest reusability value and notably P2 has high values of the DSP and PIS.

4. Empirical Study

The software used in the experiment was JFreeChart, which is a powerful and flexible open source-charting library. JfreeChart was chosen as the target software system because it is a long-term open source library with a rich set of release notes and documents to confirm the observations. An empirical study on the package metrics was done by taking sixteen packages from the three successive versions, JFree chart 0.9.21, 1.0.0 and 1.0.1. The quality attributes reusability has been studied on these packages.

The actual metric values of sixteen packages on three versions is shown in the Table 4. The first column (a) of each metric value belongs to the version 0.9.21. The second column (b) is version 1.0.0 and the third column (c) of version 1.0.1. These versions belong to the same system- JFreeChart.

Table 4. Actual Metric Values for JFreeChart Packages in Three Successive Versions, (a) version 0.9.21 (b) version 1.0.0 and (c) version 1.0.1

Package	DSP			CE			$LCOM_{avg}$			PIS		
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
org.jfree.chart	1.00	0.69	0.81	1.00	1.11	1.11	1.00	1.19	1.05	1.00	0.68	0.80
org.jfree.chart.annotations	1.00	1.40	1.50	1.00	1.25	1.50	1.00	1.52	1.28	1.00	1.40	1.50
org.jfree.chart.annotations.junit	1.00	1.50	1.40	1.00	1.00	1.00	1.00	1.56	2.04	1.00	1.38	1.63
org.jfree.chart.axis.junit	1.00	1.11	1.14	1.00	1.00	1.13	1.00	3.11	4.68	1.00	1.11	1.11
org.jfree.chart.junit	1.00	2.56	2.61	1.00	1.26	1.26	1.00	1.17	1.11	1.00	2.25	2.42
org.jfree.chart.labels	1.00	1.27	1.31	1.00	1.14	1.29	1.00	1.04	0.97	1.00	1.27	1.27
org.jfree.chart.labels.junit	1.00	1.06	1.19	1.00	1.50	2.00	1.00	1.69	2.16	1.00	1.06	1.19
org.jfree.chart.plot	1.00	1.26	1.26	1.00	1.00	1.00	1.00	1.18	1.22	1.00	1.26	1.26
org.jfree.chart.renderer.xy	1.00	0.97	1.00	1.00	1.08	1.08	1.00	1.48	1.49	1.00	1.00	1.04
org.jfree.chart.renderer.xy.junit	1.00	1.11	1.16	1.00	3.00	2.67	1.00	2.26	2.58	1.00	1.11	1.16
org.jfree.data	1.00	1.16	1.26	1.00	2.00	2.00	1.00	1.31	1.41	1.00	1.19	1.31
org.jfree.data.junit	1.00	1.50	1.88	1.00	1.00	1.00	1.00	2.63	2.06	1.00	1.50	1.75
org.jfree.data.statistics	1.00	1.14	1.14	1.00	1.00	1.00	1.00	1.24	1.26	1.00	1.14	1.14
org.jfree.data.statistics.junit	1.00	1.50	1.50	1.00	1.00	1.33	1.00	1.18	1.45	1.00	1.50	1.50
org.jfree.data.xy	1.00	0.97	1.48	1.00	1.00	1.00	1.00	0.93	0.85	1.00	1.00	1.56
org.jfree.data.xy.junit	1.00	1.57	3.86	1.00	2.00	2.50	1.00	1.57	2.00	1.00	1.57	3.86

The actual metric values of different ranges are combined in the computation of the reusability quality attribute indices given in equation *(i)*. The metric values for each version were normalized with respect to the metric's values in the first version of the system as shown in Table 5. The normalized value was obtained by dividing a metric value with the metric's value in the first version. The normalized value is required as the metric values should be specified within a specific range. Table 5 indicates that the normalized metric values are yielded in the range 0-4. The normalization is valid since it has been made out in the successive versions of the same system.

Table 5. Normalized Metric Values for JFreeChart Packages in the three Successive versions. (a) version 0.9.21 (b) version 1.0.0 and (c) version 1.0.1

Package	DSP			CE			LCOMavg			PIS		
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
org.jfree.chart	26	18	21	18	20	20	2.62	3.11	2.76	25	17	20
Org. jfree.chart. annotations	10	14	15	4	5	6	2.21	3.37	2.82	10	14	15
Org. jfree.chart.annotations. junit	8	12	14	4	4	4	0.78	1.22	1.59	8	11	13
org.jfree.chart.axis.junit	28	31	32	8	8	9	0.19	0.59	0.89	28	31	31
org.jfree.chart.junit	18	46	47	27	34	34	0.72	0.84	0.8	12	27	29
org.jfree.chart.labels	26	33	34	7	8	9	1.93	2.01	1.88	26	33	33
org.jfree.chart.labels.junit	16	17	19	2	3	4	0.45	0.76	0.97	16	17	19
org.jfree.chart.plot	35	44	44	18	18	18	2.5	2.94	3.05	35	44	44
org.jfree.chart.renderer.xy	32	31	32	13	14	14	1.82	2.69	2.72	27	27	28
org.jfree.chart.renderer.xy.junit	19	21	22	3	9	8	0.19	0.43	0.49	19	21	22
org.jfree.data	19	22	24	1	2	2	1.42	1.86	2	16	19	21
org.jfree.data.junit	8	12	15	2	2	2	0.16	0.42	0.33	8	12	14
org.jfree.data.statistics	14	16	16	5	5	5	2.57	3.18	3.25	14	16	16
org.jfree.data.statistics.junit	8	12	12	3	3	4	0.11	0.13	0.16	8	12	12
org.jfree.data.xy	29	28	43	3	3	3	1.89	1.75	1.6	27	27	42
org.jfree.data.xy.junit	7	11	27	2	4	5	0.21	0.33	0.42	7	11	27

Table 6 shows the computed values of the quality attribute reusability of sixteen packages for the three successive versions based on the normalized metric values given in Table 5.

Table 6. Computed Reusability Attributes Indices of Packages for the Successive Versions of JFreeChart

Package	Resuability 0.9.21	Reusability 1.0.0	Reusability 1.0.1
org.jfree.chart	1.00	0.71	0.79
org.jfree.chart.annotations	1.00	1.47	1.44
org.jfree.chart.annotations.junit	1.00	1.58	1.77
org.jfree.chart.axis.junit	1.00	1.63	2.01
org.jfree.chart.junit	1.00	2.38	2.48
org.jfree.chart.labels	1.00	1.24	1.21
org.jfree.chart.labels.junit	1.00	1.11	1.23
org.jfree.chart.plot	1.00	1.30	1.31
org.jfree.chart.renderer.xy	1.00	1.08	1.12
org.jfree.chart.renderer.xy.junit	1.00	0.92	1.14
org.jfree.data	1.00	1.00	1.14
org.jfree.data.junit	1.00	1.91	2.08
org.jfree.data.statistics	1.00	1.20	1.21
org.jfree.data.statistics.junit	1.00	1.55	1.53
org.jfree.data.xy	1.00	0.96	1.48
org.jfree.data.xy.junit	1.00	1.46	3.73

It has been noted from the Table 6 that there is -29% reusability decrease in package org. jfree. chart in the version 1.0.0 as compared to the version 0.9.21. Same conclusions can be made upon observing the metric values of these packages in Table 4. The value of DSP is decreased by -31% (6 classes and 3 interfaces have been removed from version 0.9.21 whereas 1 interface has been added in the version 1.0.0). Likewise, the PIS is decreased by -32%. Among all the packages, there is a 155% reusability increase in package org.jfree.data.xy.junit in the version 1.0.1 as compared to the version 1.0.0. Also, there is a 145% increase in DSP and PIS in the same package in the above version.

The high reusability is observed in 10 packages in successive release, which confirms that significant amount of functionality was added in these versions. This supports our belief that increase in DSP and PIS increase reusability.

Figure 2 shows a plot of reusability showing the expected increase in the reusability values. This infers us to reason out that the reusability should improve with the new release on the system.

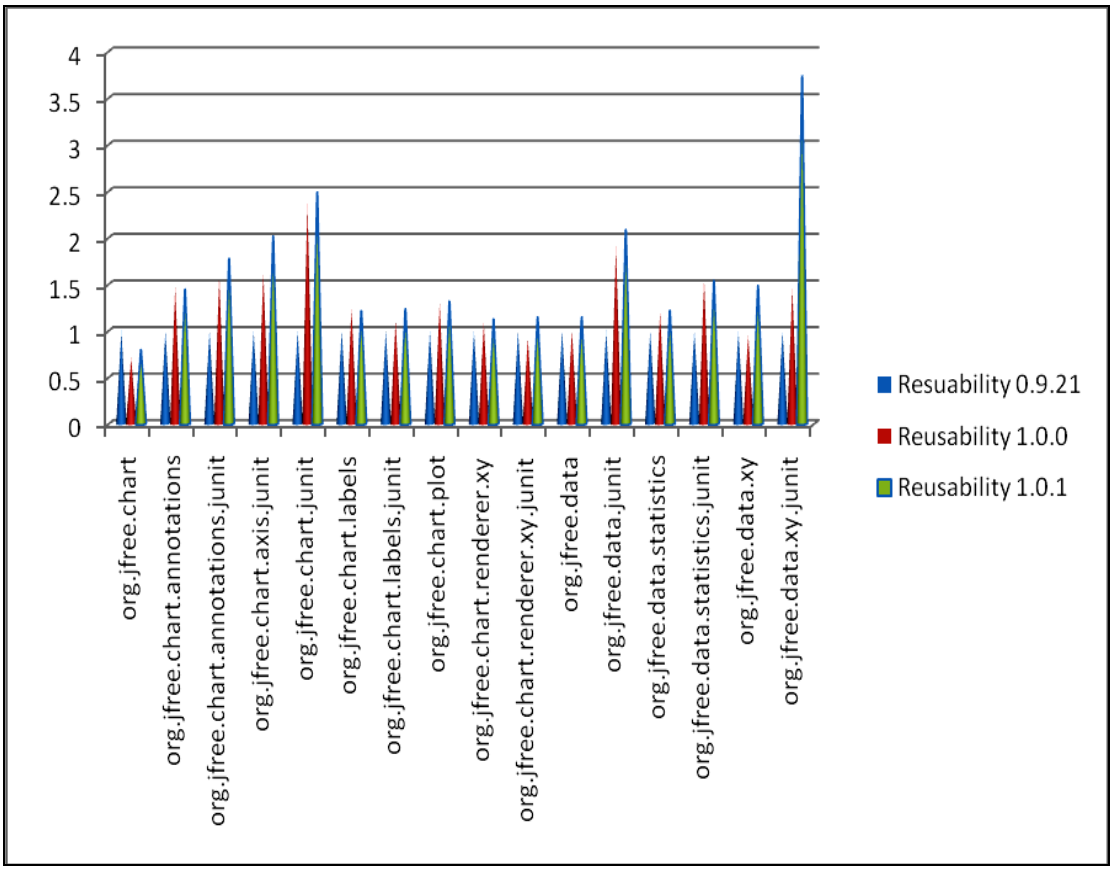


Figure 2. Plot of Computing Reusability Values of Packages for Subsequent Versions of JFree Chart

5. Conclusion

This paper is probably the first attempt to measure reusability at the package level. The reusability at the package level is significant since it is an indicator of the maintainability and testability of the new system. This paper attempts to measure the quality attribute, reusability, which is an indicator of the number of changes that occur in the successive versions. To compute the reusability, four design size package metrics namely efferent coupling (CE), cohesion ($LCOM_{avg}$) messaging (PIS) and design size (DSP) has been taken. The PIS and DSP metrics have been proposed in this paper. Metrics are then evaluated with sixteen packages of three successive versions of the same system JFree Chart. It has been observed that the package reusability gets increased in the successive versions as the new functionality gets added. It has been observed from the normalized metric values in the Table 5 and the reusability value in the Table 6 that the reusability of the packages can be calculated using the four package metrics. Also, an increase in reusability is is observed as design size and messaging increases.

References

[1] J. Bansiya and C. G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Trans. On Software Engineering, vol. 28. no. 1, **(2002)**.
[2] V. Singh and V. Bhattacherjee, "Evaluation and Application of Package Level Metrics in Assessing Software Quality", International Journal of Computer Applications (0975 – 8887), vol. 58, no. 21, **(2012)** November.

[3] R. Martin, "Agile Software Development: Principles, Patterns and practices", Prentice Hall, **(2003)**.
[4] D. Doval, S. Mancoridis and B. Mitchell, "Automatic clustering of software system using a genetic algorithm", STEP '99, IEEE computer society.
[5] T. Vernazza, G. Granatella, G. Succi and M. M. Benedicenti, "Defining metrics for software components", The World Multiconference on Systemics, Cybernetics and Informatics, Florida, **(2000)** July.
[6] S. Khan, "Design level coupling metrics for UML models", MS Thesis, KFUPM, Saudi Arabia, **(2004)** January.
[7] H. Abedeen, S. Ducasse, H. Sahraouiy and Alloui I., "Automatic package coupling and cycle minimization", WCRE '09, IEEE CNF, **(2009)**, pp. 103-122.
[8] S. A. Ebad and Ahmed., "An Evaluation Framework for Package-Level Cohesion Metrics", International Conference on Future Information Technology IPCSIT, vol. 13, **(2011)**.

Authors

Vinay Singh has received his Master of Computer Application degree from IGNOU, New Delhi, India in the year 2003 and Master of Technology in Computer science and Engineering from BITs Mesra, Ranchi, India in 2009. He is pursuing Ph.D from BIT's Mesra, India. Presently he is working as an Associate Dean of Information Technology in UMESL, Kolkata, India since 2008. He is also empanelled with Wipro Technologies as a corporate trainer. He has published twelve papers in the International Journal and Conference. His Research area is Software Metrics and Quality.

Vandana Bhattacherjee is working as a Professor, Department of Computer Science and Engineering, Birla Institute of Technology, Ranchi. She completed her B. E. (CSE) in 1989 and her M. Tech and Ph. D in Computer Science from JNU New Delhi in 1991 and 1995 respectively. She has over 100 National and International publications in Journal and Conference Proceedings. She is a member of IEEE Computer Society and Life Member of Computer Society of India. Her research areas include Software Process Models, Software Cost Estimation, Data Mining and Software Metrics.