Coupling and Cohesion Measures for Evaluation of Component Reusability

G. Gui
Department of Computer Science
University of Essex, Colchester,
CO4 3SQ, UK
Tel: +44 1206 873805
ggui@essex.ac.uk

P. D Scott
Department of Computer Science
University of Essex, Colchester,
CO4 3SQ, UK
Tel: +44 1206 872015
scotp@essex.ac.uk

ABSTRACT
This paper provides an account of new measures of coupling and cohesion developed to assess the reusability of Java components retrieved from the internet by a search engine. These measures differ from the majority of established metrics in two respects: they reflect the degree to which entities are coupled or resemble each other, and they take account of indirect couplings or similarities. An empirical comparison of the new measures with eight established metrics shows the new measures are consistently superior at ranking components according to their reusability.

Categories and Subject Descriptors
D.2.8.3 [**Metrics**]: Complexity measures.

General Terms
Measurement, Experimentation.

Keywords
Coupling, Cohesion, Reusability

1. INTRODUCTION
The work reported in this paper arose as part of a project that retrieves Java components from the internet [1]. However, components retrieved from the internet are notoriously variable in quality. It seems highly desirable that the search engine should also provide an indication of both how reliable the component is and how readily it may be adapted in a larger software system.

A well designed component, in which the functionality has been appropriately distributed to its various subcomponents, is more likely to be fault free and easier to adapt. Appropriate distribution of function underlies two key concepts: coupling and cohesion. Coupling is the extent to which the various subcomponents interact. If they are highly interdependent then changes to one are likely to have significant effects on others. Hence loose coupling is desirable. Cohesion is the extent to which the functions performed by a subsystem are related. If a subcomponent is responsible for a number of unrelated functions then the functionality has been poorly distributed to subcomponents. Hence high cohesion is a characteristic of a well designed subcomponent.

We decided that the component search engine should provide the quality rankings of retrieved components based on measures of their coupling and cohesion. There is a substantial literature on coupling and cohesion metrics which is surveyed in the next section. We then describe in detail the metrics we have developed which attempt to address some of the limitations of existing metrics. In particular, we consider both the strength and transitivity of dependencies. The following section describes an empirical comparison of our proposed metrics and several popular alternatives as predictors of reusability. Section 5 presents an analysis of the results which demonstrate that our proposed metrics consistently outperform the others. The paper concludes with a discussion of the implications of the research.

MSR'06, May 22-23, 2006, Shanghai, China.

2. COUPLING AND COHESION METRICS
Cohesion is a measure of the extent to which the various functions performed by an entity are related to one another. Most metrics assess this by considering whether the methods of a class access similar sets of instance variables. Coupling is the degree of interaction between classes. Many researches have been done on software metrics [8], the most important ones are selected used in our comparative study. Table 1 and Table 2 summarize the characteristics of these cohesion and coupling metrics.

Table 1. Coupling metrics

Name	Definition
CBO [4][5][11]	Classes are coupled if methods or instance variables in one class are used by the other. CBO for a class is number of other classes coupled with it.
RFC [4][5]	Count of all methods in the class plus all methods called in other classes.
CF [3][6]	Classes are coupled if methods or instance variables in one class are used by the other. CF for a software system is number of coupled class pairs divided by total number of class pairs.
DAC[9]	The number of attributes having other classes as their types.

Table 2. Cohesion metrics

Name	Definition
LCOM [5]	Number of non-similar method pairs in a class of pairs.
LCOM3[7][9]	Number of connected components in graph whose vertices are methods and whose edges link similar methods.
RLCOM [10]	Ratio of number of non-similar method pairs to total number of method pairs in the class.
TCC [2]	Ratio of number of similar method pairs to total number of method pairs in the class.

All of these measures have two important features in common. First, they treat relationship between a pair of classes or methods as a binary quantity; second, they treat coupling and cohesion as an intransitive relation; that is no account is taken of the indirect coupling and cohesion, although two of cohesion (LCOM3 [7][9] and TCC [2]) have suggested extensions to incorporate indirect relationships between methods. In cohesion metrics, it should be noted that three of them (LCOM, LCOM3 and RLCOM) are in fact measures of lack of cohesion. TCC [2], in contrast to the other three metrics, measures cohesion rather than its absence. In other respects it is similar to RLCOM, being the number of similar method pairs divided by the total number of method pairs.

3. PROPOSED NEW METRICS

The study suggested that none of these measures was very effective in ranking the reusability of Java components. We therefore decided to develop alternative coupling and cohesion metrics in the hope of achieving superior performance. One obvious step was to develop measures that reflected the extent to which a pair of classes was coupled or a pair of methods resembled each other. Because none of the measures treated coupling or similarity as transitive relations, we decided that such indirect dependencies should be incorporated into our metrics.

3.1 Cohesion

We develop a cohesion metric that takes account of both the degree of cohesion and transitive (i.e indirect) cohesion between methods. Methods are said to be similar if the sets of instance variables that they access overlap. We adopt a graph theoretical approach. The methods of the class are the vertices. Suppose a class has a set of method members $M \equiv \{ M_1, M_2,\ldots M_m\}$ and let. $V_j \equiv \{V_{j,1}, \ V_{j,2}, \ \ldots. \ V_{j,n}\}$ be the instance variables accessed by method M_j. Then the edge from M_j to M_i exists if and only if $V_j \cap V_i$ is not null. Thus an edge of the graph reflects the similarity of the methods in that they have at least one instance variable in common. The similarity graph is undirected because intersection is a symmetric relation. The next step is to associate a number with each edge that reflects the extent to which the two methods have instance variables in common. We therefore define $SimD(i,j)$, our measure of direct similarity of two methods, M_i and M_j, as

$$SimD\ (i,\ j) = \frac{\left|V_i \cap V_j\right|}{\left|V_i \cup V_j\right|}$$

where $i \neq j$ ($SimD(j,j)$ is defined to be zero). Note that $1 \geq SimD(i,j) \geq 0$.

The extension of the measure to include indirect similarity proceeds along the same lines as we employed for indirect coupling. The strength of similarity provided by a path between two methods is the product of the SimD values of the edges that make up the path. Thus we define $SimT(i,j,\pi)$, the transitive similarity between methods M_i and M_j due to a specific path π, as

$$SimT\ \ (i,\ j,\pi\ ) = \prod_{e_{s,t} \in \pi} SimD\ \ (s,t) = \prod_{e_{s,t} \in \pi} \frac{\left|V_s \cap V_t\right|}{\left|V_s \cup V_t\right|}$$

where $e_{s,t}$ denotes the edge between vertices s and t. As in the case of coupling, the path with the highest SimT value is selected to define the similarity of the two methods, $Sim(i,j)$.

$$Sim\ \ (i,\ j) = SimT\ \ (i,\ j,\pi_{\max}\ )$$

where $\pi_{\max}\ (i,\ j) = \arg\ \max_{\pi \in \Pi} SimT\ \ (i,\ j,\pi\ )$ and Π is the set of all paths from M_i to M_j. This measure is used to provide a measure of the cohesion of the class, *ClassCoh*, by summing the similarities of all method pairs and dividing by the total number of such pairs:

$$ClassCoh\ \ = \frac{\sum_{i,j=1}^{m} Sim\ (i,\ j)}{m^2 - m}$$

where m is the number of methods in the class. Finally, the weighted transitive cohesion of the complete software system, *WTCoh*, is defined as the mean cohesion of all the classes of which it is comprised:

$$WTCoh\ \ = \frac{\sum_{j=1}^{n} ClassCoh_j}{n}$$

where n is the number of classes in the system.

3.2 Coupling

As with cohesion measure, we regard software system as a directed graph, in which the vertices are the classes comprising the system. Suppose such a system comprises a set of classes $C \equiv \{C_1, C_2,\ldots C_m\}$. Let $M_j \equiv \{M_{j,1}, \ M_{j,2}, \ \ldots. \ M_{j,n}\}$ be the methods of the class C_j, and $R_{j,i}$ the set of methods and instance variables in class C_i invoked by class C_j for $j \neq i$ ($R_{j,j}$ is defined to be null). Then the edge from C_j to C_i exists if and only if $R_{j,j}$ is not null. Thus an edge of the graph reflects the direct coupling of one class to another. The graph is directed since $R_{j,i}$ is not necessarily equal to Ri,j.

The next step is to associate a number with each edge that reflects the extent of direct coupling from one class to another. We define $CoupD(i,j)$, as the ratio of the number of methods in class j invoked by class I to the total number of methods in class I, which indicates the impact of class j to class i.

$$CoupD\ \ (i,\ j) = \frac{\left|R_{i,j}\right|}{\left|R_i\right| + \left|M_i\right|}$$

Then the indirect coupling between classes is included. Suppose that $CoupD(i,j)$ and $CoupD(j,k)$ have finite values but that $CoupD(i,k)$ is zero. Thus although there is no direct coupling between classes C_i and C_k, there is a dependency because C_i invokes methods in C_j which in turn invokes methods in C_k. The strength of this dependency depends on the two direct couplings of which it is composed, a reasonable measure is defined as:

CoupD(*i*,*j*) × *CoupD*(*j*,*k*). This notion is readily generalised. A coupling between two classes exists if there is a path from one to the other made up edges whose *CoupD* values are all non-zero. Thus we define *CoupT*(*i*,*j*,π), the transitive coupling between classes C_i and C_j due to a specific path π, as

$$CoupT\ (i, j, \pi) = \prod_{e_{s,t} \in \pi} CoupD\ (s,t) = \prod_{e_{s,t} \in \pi} \frac{|R_{s,t}|}{|R_s| + |M_s|}$$

$e_{s,t}$ denotes the edge between vertices *s* and *t*. Note first that CoupT includes the direct coupling, which corresponds to path of length one, and second that, because the CoupD values are necessarily less than one, transitive couplings due to longer paths will typically have lower values.

In general there may be more than one path having a non-zero CoupT value between any two classes. We simply select the path with largest CoupT value and hence define Coup(i,j), the strength of coupling between the two classes, C_i and C_j to be:

$$Coup\ (i, j) = CoupT\ (i, j, \pi_{\max})$$

where $\pi_{\max}(i, j) = \arg\max_{\pi \in \Pi} CPT\ (i, j, \pi)$ and Π is the set of all paths from C_i to C_j. The final step is to use measure between each pair of classes as a basis for a measure of the total coupling of a software system. The weighted transitive coupling (*WTCoup*) of a system is thus defined

$$WTCoup = \frac{\sum_{i,j=1}^{m} Coup\ (i, j)}{m^2 - m}$$

where *m* is the number of classes in the system.

4. AN EXPERIMENTAL COMPARISON

In our study, the metrics are used for a specific purpose: predicting how much effort would be required to reuse a component within a larger system. We therefore chose to measure reusability as simply the number of lines of code that were added, modified or deleted (NLOC) in order to extend its functionality in a prescribed way. The more lines required, the lower the reusability. This appears to us to be a crude but reasonable measure of the effort that would be required to adapt a component for use within a larger system. Three case studies were carried out: *Case 1 HTML Parser*: The original components analysed HTML documents, eliminated tags and comments and output the text. The required extension was to count and output the number of tags found during parsing.

Case 2 Lexical Tokenizer: The original components tokenized a text document using user supplied token rules and output the tokens on a web interface. The required extension was to count and output the number of tokens retrieved.

Case 3 Barcode: The original components accepted a sequence of alphanumeric characters and generated the corresponding barcode. The required extension was to count the number of letters.

For each case, 20 Java components were retrieved from a repository of about 10,000 Java components retrieved form the internet. The requisite extensions were then implemented by a very experienced Java programmer and NLOC counted. Despite the relative simplicity of the extensions, there was considerable variation in the quantity of extra code required. We then proceeded to investigate how successful the various measures of coupling and cohesion are in predicting this quantity. Our proposed metrics are compared with all the metrics reviewed in section 2. In order to present the results on the same graph, those measures that do not produce values in the range (0,1) (i.e. CBO, RFC, DAC, LCOM and LCOM3) were divided by 100.

5. RESULTS

Two approaches were used to evaluate the performance of the various measures in predicting reusability: linear regression and rank correlation.

5.1 Linear Regression

The regression lines obtained for the five cohesion measures when applied to the HTML parser components are shown in Figure 1. The results for the other two sets of components were similar. It is clear that some measures provide much more consistent predictors than others. There are no obvious systematic departures from linearity so the use of simple regression appears reasonable. The regression lines obtained for coupling measures demonstrate the same situation.

The coefficient of determination, R^2, provides a measure of how much of the variation in NLOC is accounted for by the measures. Table 3 and Table 4 display the values of R^2 obtained for each of the coupling and cohesion measures on all three sets of components. In each case, our proposed new measure, WTCoup and WTCoh gave the largest value of R^2, indicating that it was the best linear predictor of reusability. The remaining measures produced at least one R^2 value so low as to indicate that that the correlation was not significantly above chance at the 5% level.

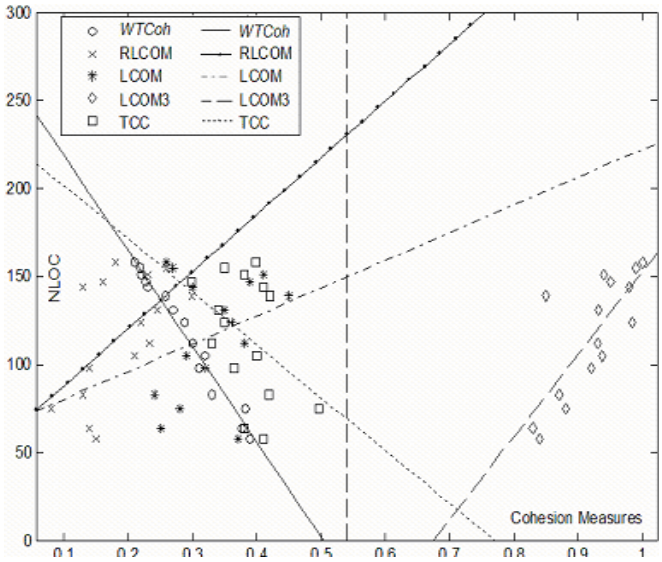

Figure 1. Regression of cohesion measures against reusability

Table 3. R^2 values for coupling measure regression lines.

Cases	WTCoup	CF	CBO	RFC	DAC
HTML Parser	.846	.621	.259	.793	.254
Lexical Token.	.836	.098	.004	.729	.738
Barcode Gen.	.958	.693	.121	534	.507

Table 4. R^2 values for cohesion measure regression lines.

Cases	WTCoh	RLCOM	LCOM3	LCOM	TCC
H. Parser	.847	.319	.259	.564	.178
L. Token.	.838	.783	.002	.709	.646
B. Gen.	.892	.702	.177	.101	.785

5.2 Spearman Rank Correlation

Although these results provide a strong indication that the proposed new measures are better predictors of reusability than the alternatives, our primary purpose is simply to rank a set of components retrieved from the repository. We therefore also computed the Spearman rank correlation coefficients between the rankings determined by NLOC and those produced by the various coupling and cohesion measures (Tables 5 and 6).

Table 5. Rank correlations values for coupling measures.

Cases	WTCoup	CF	CBO	RFC	DAC
HTML Parser	.975	.882	.465	.896	.507
Lexical Token.	.952	.291	.117	.822	.817
Barcode Gen.	.974	.758	.485	.656	.800

Table 6. Rank correlations values for cohesion measures.

Cases	WTCoh	RLCOM	LCOM3	LCOM	TCC
H. Parser	-.993	.522	.218	.564	-.343
L. Token.	.838	.783	.002	.709	.646
Bar. Gen.	.892	.702	.177	.101	.785

The relative performance of the various measures is consistent with the regression studies. In all cases, the two proposed measures, WTCoup and WTCoh, produced the highest rank correlations. They are in fact extremely high; no value was lower than 0.95.

6. DISCUSSION

These results clearly demonstrate that our proposed metrics for coupling and cohesion are very good predictors of the number of lines of code required to make simple modifications to Java components retrieved from the internet and are superior to other measures. The majority of coupling and cohesion metrics treat coupling and similarity as simple binary quantities and ignore the transitive relationship. Both our proposed measures concern these issues: First, they are weighted; that is, they use a numeric measure of the degree of coupling or similarity between entities rather than a binary quantity. Second they are transitive; that is, they include indirect coupling or similarity mediated by intervening entities. It is reasonable to enquire whether both these characteristics are necessary to achieve good prediction performance. In fact our investigations suggest that both contribute to the performance.

Although both WTCoup and WTCoh are good predictors, it is worth considering whether a linear combination might not produce even better results. Multiple regression for the Lexical Tokenizer components produced an R^2 of 0.981; the ranking produced using the regression coefficients to weight the terms had a Spearman correlation of 0.986. These are superior to the results produced by each metric alone but not by a great margin simply because there original results leave only modest scope for improvement. Developing such a composite quality measure would entail assuming the relative weighting of the two metrics should be the same for all types of component.

This work arose from, and is intended primarily as a contribution to, search engine technology. Nevertheless, we believe it may be of interest to a wider body of researchers: in particular, those involved in developing and evaluating software metrics.

7. ACKNOWLEDGMENTS

We are grateful to the four UK higher education funding bodies (for England, Scotland, Wales and Northern Ireland) for an Overseas Research Studentship (ORS/2002015010) awarded to G. Gui.

8. REFERENCES

[1] Gui, G. and Scott, P. D. Vector Space Based on Hierarchical Weighting: A Component Ranking Approach to Component Retrieval. In Proceedings of the 6th International Workshop on Advanced Parallel Processing Technologies (APPT'05)

[2] Bieman, J. M. and Kang, B-Y. Cohesion and Reuse in an Object-Oriented System. In *Proc. ACM Symposium on Software Reusability (SSR'95)*. (April 1995) 259-262.

[3] Briand, L., Devanbu, P. and Melo, W. An investigation into coupling measures for C++. *Proceedings of ICSE 1997*.

[4] Brito e Abreu, F. and Melo, W. Evaluating the impact of OO Design on Software Quality. *Proc. Third International Software Metrics Symposium.* (Berin 1996).

[5] Chidamber, S. R. and Kemerer, C. K. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, Vol. 20 (June 1994), 476-493.

[6] Harrison, R., S.J.Counsell, & R.V.Nith. An Evaluation of the MOOD Set of Object-Oriented Software Metrics. *IEEE Transactions on Software Engineering*, Vol. 24 (June 1998), 491-496.

[7] Hitz, M. and Montazeri, B. Measuring coupling and cohesion in object-oriented systems. *Proceedings of International Symposium on Applied Corporate Computing*. (Monterrey, Mexico, 1995).

[8] Kanmani, S., Uthariraj, R., Sankaranarayanan, V. and Thambidurai, P. Investigation into the Exploitation of Object-Oriented Features*. ACM Sigsoft, Software Engineering Notes*, Vol. 29 (March 2004).

[9] Li, W. & Henry, S. Object-Oriented metrics that predict maintainability. *Journal of Systems and Software*. 23(2) 1993 111-122.

[10] Li, X., Liu, Z. Pan, B. & Xing, B. A Measurement Tool for Object Oriented Software and Measurement Experiments with It. In *Proc. IWSM 2000*, 44-54.

[11] Subramanyam, R. & Krishnan, M. S. Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects. *IEEE Transactions on Software Engineering*, Vol. 29 (April 2003), 297-310.