Analysis of Software Cohesion Attribute and Test Case Development Complexity

Frank Tsui
Software Engineering
Southern Polytechnic State University
Marietta, Georgia, USA

Stanley Iriele
Software Engineering
Southern Polytechnic State University
Marietta, Georgia, USA

ABSTRACT
In this paper we report on some of our findings in our on-going research in relating Object Oriented code cohesion to test case development complexity. It is conjectured that the more cohesive the code is the easier it should be to develop test cases for the code. We studied a small ATM application software and measured the cohesion attribute using two existing cohesion metrics, LCOM5 and ITRA-C [6, 12, 14]. We then measured the test case development effort using TCD-Complexity metric [13]. Our measurements showed high correlation between ITRA-C and TCD-Complexity for OO Classes of relatively large size, but not between LCOM5 and TCD-Complexity. This result shows high potential for using certain code cohesion metric as an indicator for test complexity.

Categories and Subject Descriptors
D.2.8 [Metrics]: Complexity Measures – design complexity, testing complexity

General Terms
Measurement, Design, Verification

Keywords
cohesion metrics, design complexity, test-case complexity, correlation, measurements

1. INTRODUCTION
This is a report on the results obtained in our on-going research on various software attributes. More specifically, we are reporting on the results of studies in relating OO software cohesion attribute and software test-case development complexity.

There are numerous studies conducted on OO software cohesion [2, 3, 4, 5, 6, 7, 12, 14]. It is believed that higher cohesive software results in better software quality [4]. It is also believed that good testing results in better quality software [1, 8]. In this research, we look for relationships between the attribute of cohesion and the attribute of test case development complexity. While it is hoped that the more cohesive the code is, the easier it would be to develop test cases for that code, it is not clear that there is a close relationship between these two attributes. We studied a small ATM-processing application software which is made of nine OO Classes. The nine Classes further contained forty one Methods. Our research focuses on relating two specific OO cohesion metrics, LCOM5 [6] and ITRA-C [12], to a test-case development complexity metric, TCD-Complexity [13]. The preliminary results show some promising relationship that may lead us to cautiously use ITRA-C as a potential indicator for testing complexity and aid in projecting testing effort.

In this paper, we first explain in Section 2 the notion and the metric for a dataflow based test case development complexity, TCD complexity. In Section 3, two popular OO cohesion metrics which are close to the notion of dataflow, LCOM5 and ITRA-C are introduced. We then show in Section 4, various correlations among these measurements. The correlation results lead us to conclude in Section 5 that ITRA-C, as a OO cohesion metric, may serve as an indicator for assessing test case development complexity.

2. TESTING COMPLEXITY
In software development and software engineering, testing is one of the major tasks in the product development cycle. It is composed of a set of activities, starting from the end of requirements development until the product is integrated for delivery. Some of the main testing sub-tasks include the following:

- test case development
- test environment set up
- test execution and recording
- test result analysis

Test complexity is therefore a measurement on these sub-tasks. There has not been one universal test complexity metric that is adopted by software engineers. Neither can there be one test complexity metric to cover all these sub-tasks. However, some testing complexity issues have been addressed [9, 10, 11, 13, 15]. Of the various sub-activities in testing, test case-development may be the most challenging and time consuming. Here we will focus on the test case development task complexity. The number and type of test cases developed will dictate how much effort is needed to perform test execution and test result analysis. Thus, we will adopt a test case development task complexity metric to represent and measure the testing effort. We use the Test Case Development-Complexity or TCD-Complexity metric [13] in this research because it is based on dataflow testing technique, which in turn is close in concept to the OO cohesion metrics of LCOM5 [6] and ITRA-C [12, 14]. Dataflow testing [10, 15] is based on testing the path from data-define to data-use, known as the D-U path of the variables. The test cases developed essentially cover these D-U paths. The TCD-Complexity metric is based on the dataflow testing concept and measures the complexity involved in developing these test cases. It is defined as follows of each Object or Class:

$$\text{TCD-Complexity} = (\text{T-DU}) + |T'|$$

Furthermore, T-DU is defined as:

$$\text{T-DU} = \sum_{i=1}^{k} \sum_{j=1}^{m_i} (\text{D-U path})_{i,j} \quad \text{where}$$

- i is the ith Method
- j is the jth variable used by ith Method
- m_i is the number of variables used in ith Method and
- k is total number of Methods in the Class

There may be more than one set of test cases that will cover all the D-U paths of all the variables used in the Class. Many times the number of test cases needed to cover all of the D-U paths is considerably smaller than the number of D-U paths. |T'| in the TCD-Complexity metric represents the cardinality of the smallest set of such sets of test cases.

Note that TCD-Complexity does not only count the number of test cases. It counts the number of variables in the Methods of the Class. The amount of effort required to design the test cases requires one to study all the D-U paths originated by these variables. TCD-Complexity measures the test case development complexity, which in turn measures the effort, based on the number of D-U paths for all the variables that exist in the Methods of the Class of interest.

3. RELATING THE CONCEPTS

In this section we will further relate the concepts of cohesion to test case development and explain why we believe that there may be some interesting relationship. We use two OO cohesion metrics, LCOM5 [6] and ITRA-C [13], as the guiding post for this discussion.

With TCD-Complexity defined in terms of test case development based on dataflow testing technique, we chose the OO cohesion metrics that are conceptually close to the notion of dataflow. LCOM5 is chosen to cover the situation of common usage of instance variables by the Methods within a Class. The definition of LCOM5 is as follows.

$$\text{LCOM5} = \{ [(1/a) (\textstyle\sum u(A_j))] - m \} / (1 - m)$$

where:

a = # of attributes or instance variables
$u(A_j)$ = number of Methods that access attribute A_j
m = # of Methods in the Class
$\sum u(A_j)$ is summed over all the attributes j = 1 --- n

LCOM5 measures the relatedness of Methods in a Class, or the cohesion of a Class, by measuring how commonly the instance variables are accessed by the Methods within that class. It is based on the concept that more sharing of the instance variables by the Methods in a Class should indicate more single purposed the Class is. In a sense, the more D-U path among the instance variables and the Methods in a Class should indicate that the Class is more cohesive. Thus, one may conjecture the relationship between LCOM5 and TCD-Complexity to be such that the more cohesive the Class is, as measured by LCOM5, the more complex would test case development be in terms of TCD-Complexity. Although, we always hoped that very cohesive Classes should be easier to test and that TCD-Complexity should be lower, instead. We will investigate this relationship through measuring several sample Classes.

ITRA-C cohesion measurement is chosen for the similar reason as described above for LCOM5. ITRA-C metric also uses the variables in the Methods and considers the distance between these variables' definition and usage. It is defined as follows.

$$\text{ITRA-C (Class)} = \textstyle\sum \text{ITRA-C(Method)} / |m_j|$$

where

$|m_j|$ is the cardinality of Methods in the Class

$$\text{ITRA-C(Method)} = (E + API)/2$$

where

- API is the average Proximity indicator
- E is the inverse of Effect indicator, EI, and
- $EI = \sum \sum (v_{i,j})$ where $v_{i,j}$ represents jth variable in ith Effect code slice

We will not go into details of the Effect indicators and Proximity indicators used in ITRA-C metric here but to say that both of these indicators are based on the variables defined and used in the Methods of theClass.

Briefly, the Effect indicator is associated with the number of variables used, not only the instance variables as in LCOM5, and the Proximity indicator is associated with how far these variables are defined from where they are used. Please refer to [12, 14] for the details. Furthermore, ITRA-C uses the reciprocal of the number that is related to the number of variables. Thus it would seem that the more cohesive the Class is, as measured by ITRA-C, the less complex is the test case development as defined by TCD-Complexity. Note that this is different from the relationship expected from LCOM5 and TCD-Complexity because LCOM5 cohesion favors more D-U paths between instance variables and Methods. We will use the same sample Classes to study the ITRA-C and TCD-Complexity relationship.

Both of these OO cohesion metrics were chosen because of their conceptual basis of using instance variables and using key variables called Effects. Both of these elements are close to the notion of D-U path used in the dataflow testing techniques, which TCD-Complexity is based upon. Our expectation is that one of the cohesion metrics, LCOM5, will correlate unfavorably and the other, ITRA-C, will correlate favorably with test case development complexity as measured by TCD-Complexity.

4. CORRELATION OF LCOM5, ITRA-C AND TCD-COMPLEXITY

We have chosen a sample of nine Classes from a small ATM application to measure the LCOM5, ITRA-C, and TCD-Complexity metrics. We have also included the traditional non-commentary line of code, LOC, count as a placebo measurement. These measurements are correlated using the Spearman Rank Order Correlation Coefficient because it is clear, from metric theory, that LCOM5, ITRA-C, and TCD-Complexity are all at the ordinal measurement scale level, but it is not clear that they are at the interval or ratio scale level of measurement. Where there are ties in the ranking, we used the Pearson Correlation Coefficient Between Ranks.

The data in Table 1 shows the raw measurement results from the 9 sample Classes in the ATM application.

For LCOM5, the highest cohesion is 0 and the lowest is 1. That is, LCOM5 is defined in such a way that the smaller the number the more cohesive is the code. For ITRA-C, the range is reversed from LCOM5, but more natural. The highest cohesion is 1, and the lowest is 0. Both LOC and TCD-Complexity ranges from 1, the smallest, to some large number without a defined limit. Instead of using the raw data, the information is ranked based on the order of cohesion (high to low), order of code size (low to high), and order of test-case development complexity (low to high).

Table 1: Raw Measurement Data

Classes	LCOM5	ITRA-C	LOC (non-commentary)	TCD-Complexity
Account	.71	.78	67	34
ATM	.40	.50	142	85
BalanceInquiry	.11	.77	35	15
BankDataBase	0	.8	68	30
CashDispenser	.50	.67	32	15
Deposit	.69	.72	84	36
KeyPad	0	.72	22	6
Transaction	.75	.70	36	16
Withdrawal	.64	.66	132	65

Table 2 shows the same Classes and measurements in the ranked order form.

Table 2: Rank Ordered Data

Classes	LCOM5	ITRA-C	LOC (non-commentary)	TCD-Complexity
Account	8	2	5	6
ATM	4	9	9	9
BalanceInquiry	3	3	3	2 tie
BankDataBase	1 tie	1	6	5
CashDispenser	5	7	2	3 tie
Deposit	7	4 tie	7	7
KeyPad	2 tie	5 tie	1	1
Transaction	9	6	4	4
Withdrawal	6	8	8	8

A quick scan of the ordering shows that it is very difficult to see any clear correlation between LCOM5 and TCD-Complexity or any correlation between ITRA-C and TCD-Complexity. But LOC, the placebo measurement, and TCD-Complexity may have some correlation. Note that there are some ties in the ranking; thus Pearson Correlation Coefficient Between

Ranks is also computed. Table 3 shows the result of computing the Spearman Rank Order Correlation coefficient, r_s, and the Pearson Correlation Coefficient Between Ranks.

Table 3: Rank Order Correlation Coefficients

Correlating Attributes	Spearman Rank Order Correlation Coefficient r_s	Pearson Correlation Coefficient Between Ranks
LCOM5 vs TCD-Complexity	.32	.35
ITRA-C vs TCD-Complexity	.35	.38
LOC vs TCD-Complexity	.96	.96

For the nine Classes, both LCOM5 and ITRA-C showed low correlation to TCD-Complexity. However, the placebo measurement of LOC, correlates very highly with TCD-Complexity. After closer studies of the Classes, we noticed that the variable data usage is quite limited when the size of the Class is small. There just were not many variables in those small classes. We then limited our sample to those Classes with LOC > 50. The following 2 tables, Table 4 and Table 5, show the re-ranking of the Classes for only those with LOC> 50 and the resulting computation of Spearman Rank Correlation Coefficient for only those Classes. Note that, in Table 4, there are only 5 classes with LOC > 50. The ranking numbers for each attribute is placed in parenthesis, ( ). This time the ranking results are even more interesting. Also note that there is no tie in any of the ranking. Thus only Spearman Rank Order Correlation Coefficient, r_s, is computed in Table 5. The actual r_s for the different attribute pairs are shown in the Spearman Rank Order Correlation Coefficient table. ITRA-C showed perfect correlation with TCD-complexity. The placebo measurement of LOC continues to correlate well with TCD-Complexity. But LCOM5 showed no correlation at all with TCD-Complexity. The correlation coefficient was zero.

The results obtained with Classes whose LOC>50 made sense. The smaller Classes did not have much opportunity to show the differences between badly coded, low cohesion classes from well coded high cohesion classes. Thus, for larger Classes, the high correlation between cohesion measurement using ITRA-C and TCD-Complexity is understandable. LCOM5, on the other hand, looks at a more restrictive set of variables, instance variables. Thus, the increase in non-instance variables in large classes is not accounted for when we correlate LCOM5 with TCD-Complexity.

Table 4: For Only Classes with LOC > 50

Classes	LCOM 5	ITRA-C	LOC (non-commentary)	TCD-Complexity
*Account ***	*.71 (5)*	*.78 (2)*	*67 ** (1)*	*34 (2)*
*ATM ***	*.40 (2)*	*.50 (5)*	*142 ** (5)*	*85 (5)*
BalanceInquiry	.11	.77	35	15
*BankDataBase***	*0 (1)*	*.8 (1)*	*68 ** (2)*	*30 (1)*
CashDispenser	.50	.67	32	15
*Deposit ***	*.69 (4)*	*.72 (3)*	*84 ** (3)*	*36 (3)*
KeyPad	0	.72	22	6
Transaction	.75	.70	36	16
*Withdrawal ***	*.64 (3)*	*.66 (4)*	*132 ** (4)*	*65 (4)*

Table 5: Spearman Rank Order Coefficient, r, for Classes (LOC >50)

Correlating Attributes	r_s
LCOM5 vs TCD-Complexity	.00
ITRA-C vs TCD-Complexity	1.00
LOC vs TCD-Complexity	.90

.

While LCOM5 measures cohesion at the Class level, ITRA-C may be used to measure cohesion at both the Class level and at the Method level. Learning from the Class size measurement experience, we limited our Method sample to those with LOC>20. There were 7 of these Methods, and the following Table 6 shows the measurements.

Table 6: Measurements for Methods with LOC>20

Methods	ITRA-C	LOC	TCD-Complexity
PerformTransaction	.47 (1 –tie)	31 (3)	17 (2)
createTransaction	.37 (4)	22 (1 –tie)	24 (4 –tie)
execute	.47 (2 –tie)	22 (2 –tie)	11 (1)
Execute1	.43 (3)	41 (4)	19 (3)
Execute2	.30 (7)	58 (7)	28 (6)
DisplayMeuofAmounts	.31 (6)	45 (5)	24 (5 -tie)
Save	.33 (5)	55 (6)	47 (7)

LCOM5 is not included in Table 6 because it does not measure at the Method level. There are some ties in the ranking. Thus we again compute both Spearman's r_s and the Pearson Correlation Coefficient Between Ranks. The results of the computation, in Table 7, show that the correlation between ITRA-C and TCD-Complexity is still good at .85.

Table 7: Correlation Coefficients for Methods

	Spearman Rank Order Correlation Coefficient, r_s	Pearson Correlation Coefficient Between Ranks
ITRA-C vs TCD-Complexity	.86	.85
LOC vs TCD-Complexity	.75	.75

It continues to correlate better than LOC and TCD-Complexity when the size of the Method or Class is taken into account. Thus our preliminary set of data shows that Class cohesion as measured by ITRA-C may be a reasonably good indicator for projecting testing effort in terms of TCD-Complexity for those Classes and Methods that are not very small. More important than just LOC size is the number of variables used in those Methods and Classes.

5. CONCLUSION

This paper is a report on some results from our research on code cohesion and their relation to testing complexity. Specifically, we used two cohesion metrics, LCOM5 and ITRA-C, which utilize the data variables as the main source of measurement. We also included a placebo measurement of LOC in our research. These measurements were correlated with a test case development measurement, TCD-Complexity, which also draws on the data variables as the basis of measurement. Our results showed that LCOM5 had very low to no correlation with TCD-Complexity. ITRA-C correlates well with TCD-complexity when we eliminated low LOC samples. ITRA-C consistently beat LOC in those cases. The preliminary results show that ITRA-C, with large Classes and Methods, may serve as an indicator for assessing test case development complexity for dataflow testing effort. This will allow us to perform some early estimation of test case development effort as the code is being developed.

In our continuing research, we will need to expand our sample set to include other types of software application to see if the correlation holds. We will also need to improve our understanding of the threshold of LOC and number of variables is considered "large" enough for ITRA-C to correlate well with TCD-Complexity.

6. ACKNOWLEDGEMENTS

This research was supported in part by the Peach State Louis Stokes Alliance for Minority Participation (PSLSAMP) fund.

7. REFERENCES

[1] Amman, P. and Offut, J. 2008. *Introduction to Software Testing*, Cambridge University Press.

[2] Bieman, J. and Kang, B.K. 1998. Measuring Design Level Cohesion. *IEEE Transactions on Software Engineering,* Vol. 24, No. 2, (February 1998), 111-124.

[3] Briand, L. C., Daly, J.W., and Wust, J. 1998. A Unified Framework for Cohesion Measurement in Object Oriented Systems. *Empirical Software Engineering Journal,* Volume 3, Number 1, (March 1998), 65-117.

[4] Briand, L.C., Wust, J., Daly, J.W., and Porter, V.D. 2000. Exploring the Relationships Between Design Measures and Software Quality in Object Oriented Systems. *Journal of Systems and Software*, Vol. 51, Issue 3, (May 2000), 245-273.

[5] Chidamber,S.R. and Kemerer, C.F. 1994. A Metric Suite for Object-Oriented Design. *IEEE*

Transactions on Software Engineering, Vol. 20, No 6, (June 1994), 476-493.

[6] Henderson-Sellers, B. 1996. *Object-Oriented Metrics: Measures of Complexity*, Prentice Hall.

[7] Hitz, M. and Montazeri, B. 1995. Measuring Coupling and Cohesion in Object Oriented Systems. In *Proceedings of International Symposium on Applied Corporate Computing*, (Monterey, Mexico, 1995), 25-27.

[8] Jorgensen, P.C. 2008. *Software Testing a Craftsman's Approach, 3rd edition*, Auerbach Publications, Boca Raton, Florida.

[9] Ntafos, S.C. 1988. A Comparison of Some Structural Testing Strategies. *IEEE Transactions on Software Engineering,* Vol.14, No.6, (June 1988), 868-874.

[10] Rapps, S. and Weyuker, E.J. 1985. Selecting Software Test Data Using Data Flow Information. *IEEE Transactions on Software Engineering*, Vol. SE-11, No.4, (April 1985), 367-375.

[11] Tai, K. C. 1980. Program Testing Complexity and Test Criteria. *IEEE Transactions on Software Engineering*, Vol. SE-6, No 6, (November 1980), 531-538.

[12] Tsui, F., Bonja, C., Duggins, S. and Karam, O. 2008. An Ordinal Metric for Intra-Method Class Cohesion. In *Proceedings of IADIS Applied Computing 2008 Conference*, (Algarve, Portugal, April, 2008).

[13] Tsui, F., Karam, O., Iriele, S. 2008. *A Test Complexity Metric Based on Dataflow Testing Technique.* Internal Report, School of Computing and Software Engineering, Southern Polytechnic State University.

[14] Tsui, F., Karam, O., Duggins, S., and Bonja, C. 2009. On Inter-Method and Intra-Method Object Oriented Class Cohesion. *International Journal of Information Technologies and Systems Approach*, (January-June, 2009), 15-32.

[15] Weyuker, E.J. 1990. The Cost of Data Flow Testing: an Empirical Study. *IEEE Transactions on Software Engineering,* Vol.16, No.2, (February 1990), 121-128.