Journal of Computer Science 1(2): 137-144, 2005
ISSN 1549-3636

Exploring the Relationship between Cohesion and Complexity

[1]Cara Stein, [2]Glenn Cox and [2]Letha Etzkorn
[1]Mathematics and Computer Science Department, Edinboro University of Pennsylvania
[2]Computer Science Department, The University of Alabama in Huntsville
Sampson Gholston, Shamsnaz Virani, Phil Farrington, Dawn Utley, Julie Fortune
ISE Department, The University of Alabama in Huntsville, USA

Abstract: Many metrics have been proposed to measure the complexity or cohesion of object-oriented software. However, the complexity or cohesion of a piece of software is more difficult to capture than these metrics imply. In fact, studies have shown that existing metrics consistently fail to capture complexity or cohesion well. This study explores the reasons behind these results: cohesion is difficult to capture from syntactic elements of code, complexity is too multi-faceted to be captured by one metric and the qualities of complexity and cohesion are not independent. These factors have resulted in metrics that are purported to measure complexity or cohesion but are inadequate or misclassified. This study shows that there is overlap between some of the complexity and cohesion metrics and points to a more basic relationship between complexity and cohesion: that a lack of cohesion may be associated with high complexity.

Key words: Software Metric, Complexity, Cohesion, Object-oriented

INTRODUCTION

Software metrics can be useful to project managers and software development teams in assessing software. Metrics can be calculated automatically from source code, providing an indication of which classes may be error-prone. For example, code that is lacking in cohesion is likely to be poorly designed and therefore more error-prone [1]. Although metrics do not provide a perfect analysis of a class or function, they provide a fast, inexpensive way to get feedback on various aspects of the quality of a class. Since software follows the Pareto principle or 80/20 rule (that is, most of the errors are in a small portion of the code) [2], metrics can provide valuable information without being perfect. Metrics can be used as one source of information suggesting error-prone segments of code. Little effort is expended in collecting metrics with an automated metrics tool and if they correctly pinpoint a few modules as error-prone, there is a good chance that many of the system errors reside in those modules.

Complexity and cohesion are qualities of software that are often used to try to find error-prone modules. The reasoning is that modules that lack cohesion are poorly designed and complex modules are difficult to implement and difficult to test completely [1].Both of these conditions lead to an increased likelihood of errors in a system. Although studies have shown a relationship between complexity metrics and faults [3-7], studies have failed to show a significant relationship between cohesion metrics and faults or changeability [8-11].

One possible explanation for the lack of correlation between cohesion metrics and fault-proneness is the difficulty of measuring cohesion from source code. Briand et al. said that syntactic metrics do not capture cohesion information well [8, 10]. Allen and Khoshgoftaar [12] define cohesion as the degree of intra-module interdependence; Yourden and Constantine define it as "how tightly bound or related [a module's] internal elements are to one another" [13]. Henderson-Sellers suggests that meaning is more important to cohesion than syntax is [14]. He gives an example of two classes, Person and Car, each with high semantic and syntactic cohesion. If each person can own only one car and each car can be owned by only one person, these classes could be merged and still have a high degree of syntactic cohesion, but the Person-Car class is not cohesive conceptually [14]. Metrics based only on syntax miss semantic cohesion entirely.

Although complexity seems like an obvious concept, it is even less well defined than cohesion. As Fenton [15] and Curtis and Carleton [16] noted, no one measure can fully capture software's complexity. Curtis and Carleton observed that most complexity metrics actually measure size. In fact, the most popular complexity metrics, McCabe's Cyclomatic Complexity and Halstead's Effort, have a strong correlation with lines of code and perform no better than lines of code in predicting software quality [16]. In actuality, complexity encompasses much more than just size. Complexity is also influenced by the psychological tendencies and experience of an individual programmer: a task that is very complex to one

programmer may be very familiar and thus very easy for another [16]. Tegarden, *et al.*,[17] list three types of psychological complexity that impact programmers: problem complexity, system design complexity and procedural complexity. Gonzalez expressed complexity as a three-dimensional matrix involving three types of complexity: syntactical, functional and computational; three dimensions of complexity: length, time and level/depth; and five domains of complexity: control structure, size of code, information content, modularity and data structure [18]. One step beyond Gonzalez's information content factor, Henderson-Sellers [14] said that semantic cohesion is actually one aspect of software complexity. These assertions begin to indicate a relationship between complexity and cohesion. More generally, it is plausible that software that is less cohesive (poorly designed or containing many unrelated ideas) should be more complex (make use of a larger or more elaborate implementation, be more difficult to implement).

MATERIALS AND METHODS

To analyze the relationships between cohesion and complexity, a variety of complexity and cohesion metrics were calculated. We analyzed two types of metrics: syntactic metrics and semantic metrics. Syntactic metrics are based on code syntax. On the other hand, semantic metrics quantify the meaning of the task performed by a piece of software, using a knowledge base of domain-related concepts and keywords.

Some mathematical notation is necessary.

Let $\{C_1, C_2, \ldots, C_m\}$ be the set of m classes in a system. Let $F_a = \{F_{a1}, F_{a2}, F_{an}\}$ be the set of n member functions belonging to class C_a. Let $A_a = \{A_{a1}, A_{a2}, \ldots, A_{ao}\}$ be the set of attribute variables belonging to class C_a.

Let # be a mapping from set F_a to set A_a such that F_{ai} # A_{aj} if function F_{ai} uses attribute A_{aj} in its implementation. Let δ be a relation on set F_a such that $F_{as}\ \delta\ F_{at}$ if $\exists x \in A_a \left(F_{as} \# x \wedge F_{at} \# x\right)$, that is, if the two functions share a common attribute variable. Bieman and Kang [19] refer to this as a direct connection.

For each class C_a, let K_a be the set of domain-related keywords associated with class C_a and let O_a be the set of domain-related concepts associated with C_a. The domain-related concepts and keywords are part of the calculation of semantic metrics [20, 21]; in calculating these metrics, concepts and keywords are associated with classes and functions using a knowledge base of domain information. Define $I_a = K_a \cup O_a$; we will refer to this as the set of ideas associated with class C_a. Let R_a be the set of conceptual relations (as defined by Sowa [22]) connected from any concept in O_a to any other concept in the knowledge base.

For each member function F_{ai}, let K_{ai} be the set of keywords associated with F_{ai}, let O_{ai} be the set of concepts associated with F_{ai} and let $I_{ai} = K_{ai} \cup O_{ai}$ be the set of ideas associated with F_{ai}.

Let O be the set of all concepts in the knowledge base. Then define mapping % from set R_a to set O such that for any r in R_a and any x in O, r%x if and only if r forms a connection from concept x to any concept y in set O or r forms a connection from any concept z in set O to concept x. Let $\rightarrow$ be a relation between concepts p and q in set O such that $p \rightarrow q$ if and only if there exists a conceptual relation from p to q. Let q_i be the number of times idea $i \in I_a$ is inferenced.

Let $S^*_a = [s_1, s_2, \ldots, s_x]$ be the bag of x name strings used in the definition of class C_a, where a name string is any user-defined identifier name, such as a class name, variable name, function name, or parameter name. Let S_a be the set of unique elements in bag S^*_a. Let u_{aj} be the frequency with which name string s_j occurs in the definition of class C_a. Similarly, let $S^*_{ai} = [s_1, s_2, \ldots, s_y]$ be the bag of y name strings used in the definition of member function F_{ai} of class C_a. Then S_{ai} is the set of unique elements in bag S^*_{ai} and u_{aik} is the frequency with which string s_k occurs in the implementation of function F_{ai}.

Let M_{aj} be the McCabe's Cyclomatic Complexity of function F_{aj}.

Using these definitions, the metrics are defined in Table 1.

Values for the metrics were calculated from three graphical user interface (GUI) systems written in C++: Gina [23], wxWindows [24] and Watson [25]. Gen++ [26] was used to calculate WMC, DIT and LCOM; HYSS [27] was used to calculate LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, LCC and TCC; Cantata++ [28] was used to calculate McCabe WMC, Bansiya CDE and Bansiya CIE; and semMet [29] was used to calculate CDC, SCDEa, LORM and PSI.

To provide a basis for comparison, two teams of experts also analyzed the same software packages and rated the cohesion and complexity of each class. Expert Team 1 consisted of seven software developers, each with five to fifteen years' experience in software development and at least three years' experience with C++ and GUI programming. Each expert had a BS in computer science or electrical engineering and all but one had a master's degree. These experts analyzed a set of 17 classes from the Gina and wxWindows systems chosen to make a minimal windowing system [30]. Expert Team 2 consisted of the students in a graduate-level software engineering course. Most of these experts had at least a year of experience in software development and all had prior object-oriented programming experience, especially C++. This team analyzed 13 classes taken from the wxWindows system to make a minimal windowing system [30].

Table 1: Metric Definitions

Metric	Definition	Source
Semantic Complexity Metrics		
CDC	$\sum_{i \in O_a} \left(\left(1 + \left\| \{ x \mid i \rightarrow x \wedge x \in O_a \} \right\| \right) * w_i \right)$, where w_i is the weighting factor for concept i	[20]
SCDEa	SCDEa = $-\sum_{i \in K_a} \left(\frac{q_i}{\sum_{j \in K_a} q_j} \log_2 \left(\frac{q_i}{\sum_{j \in K_a} q_j} \right) \right)$	[30]
Syntactic Complexity Metrics		
WMC	WMC= $\left\| F_a \right\|$ (as calculated by Gen++)	[33, 34]
McCabe WMC	McCabe WMC = $\sum_{i=1}^{\|F_a\|} M_{ai}$	[33, 35]
Bansiya CDE	CDE = $-\sum_{j=1}^{\|S^*_a\|} \left(\frac{u_{aj}}{\|S_a\|} \log_2 \left(\frac{u_{aj}}{\|S_a\|} \right) \right)$	[36, 37]
Bansiya CIE	CIE=- $\sum_{i=1}^{\|F_a\|} \sum_{j=1}^{\|S^*_{ai}\|} \left(\frac{u_{aij}}{\|S_{ai}\|} \log_2 \left(\frac{u_{aij}}{\|S_{ai}\|} \right) \right)$	[36]
DIT	The depth of the class in the inheritance hierarchy. In the case of multiple inheritance, choose the largest of these values for the class.	[33, 34]
Semantic Cohesion Metrics		
LORM	$\frac{\left\| \{ (F_{ax}, F_{ay}) \mid F_{ax}, F_{ay} \in F_a \wedge \exists r \in R_a \left(r\%p \wedge r\%q \wedge p \in I_{ax} \wedge p \notin I_{ay} \wedge q \in I_{ay} \wedge q \notin I_{ax} \right) \} \right\|}{\frac{\|F_a\|(\|F_a\|-1)}{2}}$	[20]
PSI	$\frac{\left\| \{ x \mid \exists (i,j) \left(x \in I_{ai} \wedge x \in I_{aj} \right) \} \right\|}{\left\| \{ y \mid \exists k \left(y \in I_{ak} \right) \} \right\|}$ for $1 \le i, j, k \le \|F_a\|$, or 0 if $\forall i \left(I_{ai} = \varnothing \right)$	[21]
Syntactic Cohesion Metrics		
LCOM	$P = \{ (x,y) \mid x,y \in F_a \wedge \forall A_{ai} \left(\neg x \# A_{ai} \vee \neg y \# A_{ai} \right) \}$ LCOM = |P|	[34]
LCOM1	$P' = \{ (x,y) \mid x,y \in F_a \wedge x \neq y \wedge \forall A_{ai} \left(\neg x \# A_{ai} \vee \neg y \# A_{ai} \right) \}$ LCOM1 = |P'|	[38]
LCOM2	$P' = \{ (x,y) \mid x,y \in F_a \wedge x \neq y \wedge \forall A_{ai} \left(\neg x \# A_{ai} \vee \neg y \# A_{ai} \right) \}$ $Q' = \{ (x,y) \mid x,y \in F_a \wedge x \neq y \wedge \exists A_{ai} \left(x \# A_{ai} \wedge y \# A_{ai} \right) \}$ LCOM2 = |P'| - |Q'|, or 0 if |Q'| |P'|	[35]
LCOM3	$G_a = (V_a, E_a)$ is an undirected graph with vertices $V_a = F_a$ and edges $E_a = \{ (x,y) \mid x,y \in F_a \wedge \exists A_{ai} \left(x \# A_{ai} \wedge y \# A_{ai} \right) \}$ LCOM3 = the number of connected components of G_a	[39, 40]
LCOM4	$G_a' = (V_a, E_a)$ is an undirected graph with vertices $V_a = F_a$ and edges $E_a = \{ (x,y) \mid x,y \in F_a \wedge \exists A_{ai} (x \# A_{ai} \wedge y \# A_{ai}) \} \cup \{ (w,z) \mid w,z \in F_a \wedge w \mapsto z \}$ LCOM4=the number of connected components of G_a'	[39]
LCOM5	$\frac{\|F_a\| - \frac{1}{\|A_a\|} \sum_{i=1}^{\|A_a\|} \left\| \{ x \mid x \in F_a \wedge x \# A_{ai} \} \right\|}{\|F_a\| - 1}$	[38]
LCC	LCC= $\frac{\left\| \{ (F_{ax}, F_{ay}) \mid (F_{ax} \delta F_{ay}) \vee (\exists F_{ai}, F_{aj}, \ldots, F_{an} \left(F_{ax} \delta F_{ai} \wedge F_{ai} \delta F_{aj} \wedge \ldots \wedge F_{an} \delta F_{ay} \right)) \} \right\|}{\frac{\|F_a\|(\|F_a\|-1)}{2}}$	[19]
TCC	$\frac{\left\| \{ (F_{ai}, F_{aj}) \mid F_{ai} \delta F_{aj} \} \right\|}{\frac{\|F_a\|(\|F_a\|-1)}{2}}$	[9]

Table 2: Metric Correlations with Expert Cohesion Ratings

	Expert Team 1			Expert Team 2		
Metric	Correlation	p-value	Statistically Significant (=0.10)	Correlation	p-value	Statistically Significant (=0.10)
Semantic Complexity Metrics						
CDC	-0.71	0.002	✓	-0.93	<0.001	✓
SCDEa	-0.67	0.005	✓	-0.83	0.001	✓
Syntactic Complexity Metrics						
WMC	-0.80	0.004	✓	-0.87	0.001	✓
McCabe WMC	-0.62	0.013	✓	-0.82	0.001	✓
Bansiya CDE	-0.87	<0.001	✓	-0.73	0.005	✓
Bansiya CIE	-0.86	<0.001	✓	-0.71	0.007	✓
DIT	-0.03	0.913		0.45	0.119	
Semantic Cohesion Metrics						
LORM	-0.64	0.008	✓	-0.90	<0.001	✓
PSI	-0.28	0.291		-0.77	0.003	✓
Syntactic Cohesion Metrics						
LCOM	-0.44	0.104		-0.58	0.062	✓
LCOM1	-0.51	0.043	✓	-0.51	0.094	✓
LCOM2	-0.50	0.048	✓	-0.48	0.112	
LCOM3	-0.50	0.046	✓	-0.39	0.214	
LCOM4	-0.51	0.044	✓	-0.46	0.137	
LCOM5	-0.40	0.157		-0.41	0.239	
LCC	0.40	0.202		0.49	0.155	
TCC	-0.25	0.349		0.04	0.893	

In order to assess the relationships between complexity and cohesion metrics, we perform statistical tests using correlation. Correlation values range from -1.0 to 1.0. To help understand correlations, Cohen [31] and Hopkins [32] proposed the following scale for correlation magnitude:

<0.1 trivial
0.1-0.3 minor
0.3-0.5 moderate
0.5-0.7 large
0.7-0.9 very large
0.9-1.0 almost perfect

For each experiment, Pearson's correlation coefficient was used. The null and alternate hypotheses were:

H_0: There is no correlation between the two variables (=0)
H_1: There is a correlation between the two variables (0)

In these experiments, rejecting the null hypothesis indicates that there is a statistically significant relationship between the two quantities being studied, for example, a metric and an expert team's assessment of cohesion.
Since the metrics being studied are calculated at the class level (that is, each metric produces one value per class in an object-oriented software system), the experimental unit for each experiment is a class.
We present correlations among metrics and experts' ratings of complexity and cohesion, showing considerable crossover between the two groups of metrics. Then we present a principal component analysis of a large set of complexity and cohesion metrics, showing that all of the complexity, cohesion and unclassified metrics studied measure a total of only five different qualities of software.

RESULTS

Metrics vs. Expert Cohesion Ratings: First we compared the complexity and cohesion metrics to the experts' assessments of cohesion. Table 2 shows the results of this analysis. The table shows that there is a very strong negative correlation between the expert's assessment of cohesion and most of the complexity metrics (CDC, SCDEa, WMC, McCabe WMC and Bansiya CDE and CIE). In fact, many of the complexity metrics actually correlate more strongly with the expert assessments of cohesion than do the cohesion metrics.
For Expert Team 1, all of the complexity metrics, with the exception of McCabe WMC and DIT, correlated better with the expert assessment than did any of the cohesion metrics. The best correlation was found for Bansiya's CDE and CIE complexity metrics, which had

Table 3: Metrics vs. Expert Complexity Ratings

	Expert Team 1			Expert Team 2		
Metric	Correlation	p-value	Statistically Significant (=0.10)	Correlation	p-value	Statistically Significant (=0.10)
Semantic Complexity Metrics						
CDC	-0.53	0.033	✓	-0.67	0.018	✓
SCDEa	-0.55	0.029	✓	-0.75	0.005	✓
Syntactic Complexity Metrics						
WMC	-0.18	0.562		-0.31	0.310	
McCabe WMC	-0.51	0.050	✓	-0.71	0.007	✓
Bansiya CDE	-0.50	0.058	✓	-0.65	0.016	✓
Bansiya CIE	-0.56	0.031	✓	-0.71	0.006	✓
DIT	-0.18	0.524		-0.05	0.866	
Semantic Cohesion Metrics						
LORM	-0.64	0.008	✓	-0.68	0.015	✓
PSI	-0.19	0.478		-0.38	0.225	
Syntactic Cohesion Metrics						
LCOM	-0.03	0.929		-0.16	0.642	
LCOM1	-0.33	0.212		-0.45	0.138	
LCOM2	-0.34	0.200		-0.45	0.144	
LCOM3	-0.52	0.040	✓	-0.58	0.046	✓
LCOM4	-0.61	0.013	✓	-0.63	0.029	✓
LCOM5	-0.47	0.088	✓	-0.24	0.496	
LCC	-0.19	0.560		-0.10	0.791	
TCC	-0.28	0.286		-0.44	0.149	

correlation coefficients of -0.86 and -0.87, respectively. By comparison, the strongest correlation between a cohesion metrics and the experts was -0.64 for the LORM semantic metric.

Similar results were obtained for Expert Team 2. The CDC complexity metric had a nearly perfect correlation (-0.93) with the expert assessments of cohesion, slightly better than the best cohesion metric, LORM (-0.90). In fact, all of the complexity metrics had a better correlation with the experts than any of the cohesion metrics, with the exception of the two semantic cohesion metrics, LORM and PSI.

These findings indicate that there is a strong negative correlation between the quantities measured by complexity metrics and the cohesion of a class. Although this idea is not obvious, it is plausible that the more complex a class is, the more tasks it includes, so there is an increased likelihood that some of those tasks are unrelated causing reduced cohesion.

To investigate one example, the CDC and SCDE metrics can be considered. Both metrics are based on the number of ideas in a class. CDC counts the ideas and multiplies them by a weighting factor; SCDE uses the number of ideas and how many times they were inferenced. In light of this, it is plausible that the CDC and SCDE metrics would be measures of lack of cohesion (indicated by the negative correlation). The more ideas are in a class, the higher these metrics' values will be. At the same time, the more ideas are in a class, the less likely it is that the class accomplishes only one unified task in the domain. The implication is that the larger the number of ideas that are present in a class, the less cohesive the class is likely to be. This is the relationship shown in the results of this experiment.

However, this does not explain the strong relationship between WMC and the experts' cohesion ratings. WMC in this experiment is the version calculated by Gen++ [26]: each method is assumed to have a complexity of one. Therefore, this version of WMC is the same as simply counting the member functions in a class.

Combining the findings from the semantic metrics and WMC, we see that a class containing more ideas and more functions is less cohesive. This conclusion is consistent with the definition of cohesion.

Metrics vs. Expert Complexity Ratings: Given the remarkable performance of the semantic complexity metrics as measures of lack of cohesion, we wondered if there might be a correspondingly large relationship between the cohesion metrics and the experts' assessment of complexity. Results of correlating the two sets of data (Table 3) do show a few significant correlations (it should be noted that because the expert teams rated complexity on a decreasing scale -- "not complex" = 1.00, "fairly complex" = 0.50 and "very complex" = 0.00 --conventional complexity metrics have an negative correlation with the expert results.) For example, the LORM and LCOM4 cohesion metrics correlate with the experts' complexity assessments with correlation coefficient magnitudes in the range 0.6 to

0.7. However, the majority of the cohesion metrics show only weak correlation with the expert complexity assessments.

This result indicates that the implication of the first analysis – that if the complexity metric of a class is high (low), the class will have a low (high) cohesion – cannot be reversed. That is, high (low) cohesion metrics do not imply low (high) complexity. This result is reasonable in light of the fact that low class cohesion can result from many factors that do not relate to high complexity. For example, if a class contains only two functions that are not related, the cohesion will be low, while most complexity assessments would show low complexity due to the small size of the class.

Pair-wise Correlation of Cohesion and Complexity Metrics: To further investigate the relation between complexity metrics and cohesion, we performed a pair-wise correlation of the cohesion and complexity metrics. For this analysis, the metric values were calculated for classes taken from the wxWindows and Gina GUI suites. The number of classes used to calculate the individual metrics ranged from 34 to 360 with a mean of 137 classes.

Table 4: Cohesion Metrics Correlated with Complexity Metrics

		Complexity						
		CDC	SCDEa	WMC	McCabe WMC	Bansiya CDE	Bansiya CIE	DIT
Cohesion	LORM	0.30	--	0.27	--	0.29	0.33	--
	PSI	0.45	--	0.55	--	0.48	0.49	--
	LCOM	0.15	--	0.44	--	0.34	0.30	--
	LCOM1	0.26	--	0.77	0.31	0.38	0.41	--
	LCOM2	0.25	--	0.61	--	0.29	0.34	--
	LCOM3	0.34	--	0.62	--	0.41	0.48	0.27
	LCOM4	0.36	--	0.52	--	0.35	0.43	0.28
	LCOM5	--	--	0.27	--	0.54	0.48	--
	LCC	--	--	--	--	--	--	0.43
	TCC	-0.17	--	--	--	--	--	0.40

Table 5: Cohesion Metrics Correlated with Cohesion Metrics

		Cohesion									
		LORM	PSI	LCOM	LCOM1	LCOM2	LCOM3	LCOM4	LCOM5	LCC	TCC
Cohesion	LORM		0.33	0.14	--	--	0.11	0.14	--	--	--
	PSI	0.33		0.16	0.40	0.39	0.48	0.50	--	--	0.28
	LCOM	0.14	0.16		0.35	0.28	0.14	--	--	--	--
	LCOM1	--	0.40	0.35		0.99	0.84	0.51	--	0.33	0.39
	LCOM2	--	0.39	0.28	0.99		0.86	0.53	--	0.29	0.30
	LCOM3	0.11	0.48	0.14	0.84	0.86		0.86	--	0.32	0.43
	LCOM4	0.14	0.50	--	0.51	0.53	0.86		--	0.26	0.40
	LCOM5	--	--	--	--	--	--	--		-0.48	0.34
	LCC	--	--	--	0.33	0.29	0.32	0.26	-0.48		0.96
	TCC	--	-0.22	-0.15	--	--	--	--	--	0.96	

The results of the pair-wise correlation analysis between the cohesion and complexity metrics are shown in Table 4 (for comparison, the results of correlating cohesion and complexity metrics with others of their own type are shown in Tables 5 and 6). The values shown are those where results are statistically significant (α=0.10); non-statistically-significant results are indicated by "--".

Table 6: Complexity Metrics Correlated with Complexity Metrics

		Complexity						
		CDC	SCDEa	WMC	McCabe WMC	Bansiya CDE	Bansiya CIE	DIT
Complexity	CDC		0.56	0.49	--	0.53	0.55	--
	SCDEa	0.56		0.43	--	0.55	0.56	0.30
	WMC	0.49	0.43		0.24	0.80	0.77	-0.41
	McCabe WMC	--	--	0.24		0.18	--	-0.41
	Bansiya CDE	0.53	0.55	0.80	0.18		0.95	0.18
	Bansiya CIE	0.55	0.56	0.77	--	0.95		0.33
	DIT	--	0.30	-0.41	-0.41	0.18	0.33	

Table 7: Principal Components

		Principal Component				
		1	2	3	4	5
Cohesion	LORM	X				
	PSI	X				
	LCOM					X
	LCOM1					X
	LCOM2			X		
	LCOM3			X		
	LCOM4			X		
	LCOM5		X			
	LCC		X			
	TCC		X			
Complexity	CDC	X				
	SCDEa	X				
	WMC	X				
	McCabe WMC				X	
	Bansiya CDE	X				
	Bansiya CIE	X				
	DIT				X	

As shown in Table 4, there are several instances in which a cohesion and complexity metric correlate at a "moderate" level or above. The WMC metric, in particular, correlates well with the majority of the cohesion metrics (PSI, LCOM1-4). There is also good correlation between the Bansiya CDE complexity metric and the LCOM5 cohesion metric. (It should be noted that the cohesion metrics analyzed here are "Lack of Cohesion" metrics, having values that increase with decreased cohesion. Thus, positive correlation with the complexity metrics is consistent with the results in the previous section).

The pair-wise analysis shows that the concept of a negative correlation between complexity metrics and cohesion extends to some significant cohesion *metrics*.

Principal Component Analysis: Principal Component Analysis (PCA) is a statistical technique that categorizes variables into groups based on the similarity of what they measure. For instance, a metrics study may include 50 complexity metrics, but those metrics may only be measuring three different aspects of complexity: psychological, control structure and size. Each group, or Principal Component (PC), is said to measure a different orthogonal dimension of the entity being measured [10].

To further investigate commonalities between the cohesion and complexity metrics, we conducted a Principal Components Analysis. The PCA identified the five Principal Components shown in Table 7. The most important PC for this study is PC1 which includes two of the cohesion metrics (LORM and PSI) as well as the majority of the complexity metrics, The PCs indicate that:

* LCOM5, LCC, TCC measure one kind of value, whereas
* LCOM2-4 measure another kind of value and
* LCOM1-2 measure yet another kind of value, while
* McCabe WMC and DIT measure yet another kind of value

LORM, PSI, CDC, SCDEa, WMC, Bansiya CDE and Bansiya CIE appear to measure similar features.

The PC analysis shows that, consistent with our earlier studies, complexity metrics can be used to measure cohesion. The PC analysis further reveals that the complexity metrics are measuring the same kind of cohesion as the semantic cohesion metrics, LORM and PSI. However, the complexity metrics are measuring a different kind of cohesion than is measured by LCOM5, TCC, LCC, LCOM1-4.

We note that McCabe WMC and DIT are also measuring different features than are measured by the other complexity metrics, WMC, Bansiya CDE and CIE, CDC and SCDEa.; we also note that different cohesion metrics seem to be measuring different kinds of values. Further study is focusing on determining the common factors addressed by the various metrics.

DISCUSSION

Our results support the idea that there is a relationship between complexity and cohesion, basically that a lack of cohesion is associated with high complexity. Some metrics are clearly complexity metrics; cohesion metrics are much less concrete in their classification and some metrics appear to be incorrectly classified. We hope this study will provide a better understanding of complexity and cohesion as measured by metrics and the relationship between them.

Although many have asserted that it is impossible for one number or metric value to capture all aspects of complexity, we feel that the approach taken by Gonzalez [18] in using a model that combines many aspects of complexity is a step in the right direction. Perhaps new metrics that combine aspects of syntactic and semantic complexity and cohesion will perform better in predicting fault-proneness, effort and changeability than current metrics.

ACKNOWLEDGEMENTS

The research in this study was partially supported by NASA grants NAG5-12725 and NCC8-200.

REFERENCES

1. Pressman, R., 2001. Software Engineering. 5th ed. McGraw-Hill, Boston.
2. Fenton, N. and N. Ohlsson, 2000. Quantitative analysis of faults and failures in a complex software system. IEEE Transactions on Software Engineering, 26: 797-814.
3. Basili, V., L. Briand and W. Melo, 1996. A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, 22: 751-761.
4. French, V., 1995. Applying software engineering and process improvement to legacy defense system maintenance: an experience report. Proceedings of the 6th International Symposium on Software Reliability Engineering, pp: 34-39.
5. Khoshgoftaar, T. and J. Munson, 1990. Predicting software development errors using software complexity metrics. IEEE J. on Selected Areas in Communications, 8: 253-261.
6. Ohlsson, N. and H. Alberg, 1996. Predicting fault-prone software modules in telephone switches. IEEE Transactions on Software Engineering, 22: 886-894.
7. Subramanyam, R. and M. Krishnan, M., 2003. Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects. IEEE Transactions on Software Engineering, 29: 297-310.
8. Briand, L., J. Daly, V. Porter and J. Wust, 1998. A comprehensive empirical validation of design measures for object-oriented systems. Proceedings of the 5th International Software Metrics Symposium, pp: 246-257.
9. Briand, L., J. Daly, V. Porter and J. Wust, 1998. Predicting fault-prone classes with design measures in object-oriented systems. Proceedings of the 9th International Symposium on Software Reliability Engineering, pp: 334-343.
10. Briand, L., J. Wust, J. Daly and V. Porter, 2000. Exploring the relationships between design measures and software quality in object-oriented systems. J. Systems and Software, 51: 245-273.
11. Kabaili, H., R. Keller and F. Lustman, 2001. Cohesion as changeability indicator in object-oriented systems. Proceedings of the 5th European Conference on Software Maintenance and Reengineering, pp: 39-46.
12. Allen, E. and T. Khoshgoftaar, 1999. Measuring coupling and cohesion: an information theory approach. Proceedings of the 6th Annual Software Metrics Symposium, pp: 119-127.

13. Yourdon, E. and L. Constantine, 1979. Structured Design. Prentice Hall, Englewood Cliffs, NJ.
14. Henderson-Sellers, B., 1996. Object-Oriented Metrics. Prentice-Hall PTR, Upper Saddle River, NJ.
15. Fenton, N., 1994. Software measurement: a necessary scientific basis. IEEE Transactions on Software Engineering, 20: 199-206.
16. Curtis, B. and A. Carleton, 1994. Seven ± two software conundrums. Proceedings of the 2nd International Metrics Symposium, pp: 96-105.
17. Tegarden, D., S. Sheetz and D. Monarchi 1995. A software complexity model of object-oriented systems. Decision Support Systems, 13: 241-262.
18. Gonzalez, R., 1995. A unified metric of software complexity: measuring productivity, quality and value. Journal of Systems Software, 29: 17-37.
19. Bieman, J. and B.K. Kang, 1995. Cohesion and Reuse in an Object-Oriented System. Proceedings of the ACM Symposium on Software Reusability, pp: 259-262.
20. Etzkorn, L. and H. Delugach, 2000. Towards a Semantic Metrics Suite for Object-oriented Design. Proceedings of the 34th International Conference on Technology of Object-Oriented Languages and Systems, pp: 71-80.
21. Stein, C., L. Etzkorn, G. Cox, P. Farrington, S. Gholston, D. Utley and J. Fortune, 2004. A New Suite of Metrics for Object-Oriented Software. Proceedings of the 1st International Workshop on Software Audit and Metrics, pp: 49-58.
22. Sowa, J., 1984. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, Mass.
23. Backer, A., A. Genau and M. Sohlenkamp, 2004 (last accessed). The Generic Interactive Application for C++ and OSF/MOTIF, version 2.0. Anonymous ftp at ftp.gmd.de, directory gmd/ginaplus.
24. Smart, J. 2004 (last accessed). wxWindows. http://www.wxwindows.org.
25. Watson, M., 1993. Portable GUI Development with C++. McGraw-Hill, New York.
26. Devanbu, P., 1992. GENOA - A customizable, language- and front-end independent code analyzer. Proceedings of the International Conference on Software Engineering, pp: 307-317.
27. Chae, H., Y. Kwon and D. Bae. A Cohesion Measure for Object-Oriented Classes. Software: Practice and Experience, 30: 1405-1431.
28. Information Processing Ltd., 2004 (last accessed). Cantata++ Technical Brief. http://www.qcsltd.com/cantpp/canpp_tb.pdf.
29. Stein, C., L. Etzkorn, S. Gholston, D. Utley and G. Cox, 2004. Early Software Qualification through Semantic Program Understanding. Submitted to Intl. J. Computers and Applications.
30. Etzkorn, L., S. Gholston and W. Hughes, 2002. A Semantic Entropy Metric. J. Software Maintenance and Evolution, 14: 293-310.
31. Cohen, J., 1998. Statistical Power Analysis for Behavioral Science. 2nd ed. Lawrence Erlbaum Publishing, Mahwah, NJ.
32. Hopkins, W. 2004 (last accessed). A New View of Statistics. http://www.sportsci.org/resource/stats.
33. Chidamber, S. and C. Kemerer, 1994. A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 20: 476-493.
34. Chidamber, S. and C. Kemerer, 1991. Towards a Metrics Suite for Object Oriented Design. Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications, pp: 197-211.
35. McCabe, T. A Complexity Measure, 1976. IEEE Transactions on Software Engineering, SE-2: 308-320.
36. Bansiya, J., C. Davis and L. Etzkorn, 1999. An Entropy-Based Complexity Measure for Object-Oriented Designs. Theory and Practice of Object Systems, 5: 111-118.
37. Etzkorn, L., J. Bansiya and C. Davis, 1999. Design and Complexity Metrics for OO Classes. J. Object-oriented Programming, 12: 35-40.
38. Henderson-Sellers, B., 1996. Software Metrics. Prentice Hall, Hemel Hempstead, UK.
39. Hitz, M. and B. Montazeri, 1995. Measuring Coupling and Cohesion in Object-Oriented Systems. Proceedings of the International Symposium on Applied Corporate Computing.
40. Li, W. and S. Henry, 1993. Object-Oriented Metrics that Predict Maintainability. J. Systems Software, 23: 111-122.