Experimental evaluation of agreement between programmers in applying the rules of cohesion

Jagadeesh Nandigam[*], Arun Lakhotia[†], and Claude Cech[‡]

[*]Dept. of Math and Computer Science
Grambling State University
Grambling, LA 71245
Email: jagadeesh@vax0.gram.edu

[†]The Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504
(318) 482-6766, -5791 (Fax)
arun@cacs.usl.edu

[‡]Psychology Department
University of Southwestern Louisiana
Lafayette, LA 70504
Email: cgc2646@usl.edu

Please direct all correspondence to the second author.

Abstract

In the early 1970s Stevens, Myers, Constantine, and Yourdon introduced the notion of module cohesion and presented rules that could be used to assess the cohesion of a module. Their rules consisted of a set of relations ordered to constitute levels, and the criterion that a module was assigned the lowest of the levels of the relations it satisfied. Stevens et al.'s rules of cohesion are now covered in most software engineering textbook, even though they have never been subjected to any experimental analysis. This paper presents the results of an experiment analyzing these rules of cohesion. The experiment, using fifteen computer science graduate students as subjects, was conducted to assess whether Stevens et al.'s rules were objective, i.e., whether there is above-chance agreement in the cohesion levels assigned by different programmers. The data collected indicates that, even though the subjects were assessed to have understood the concepts well, there is a significant variation in the cohesion levels assigned by them. This result raises questions about the precision of the material taught in the software engineering curriculum.

Index terms: Software metrics, software measures, cohesion, experimentation in software engineering, experiment design.

1 Introduction

There are two fundamental principles in engineering design: (a) maximize the functional relatedness of each component and (b) minimize the interdependencies between components. In early 1970s, Stevens, Yourdon, and Constantine observed that these principles were also relevant for designing software [22]. Stevens et al. called the interconnections between modules (commonly called procedures) *coupling* and the functional relatedness of the activities performed by a module as its *cohesion*. Based on their interviews with software designers and personal experiences, they identified several different types of relations of coupling and cohesion. They further rank ordered each of the two sets of relations in decreasing order of preference of a designer, cohesion relations were ordered high to low, and coupling low to high. Stevens et al. claimed that system designs leading to modules with higher level of cohesion and lower level of coupling were qualitatively better.

This paper relates only to the notion of cohesion. Like several other concepts in software engineering, Stevens et al.'s cohesion (and coupling) relations, have been accepted by the software engineering researchers and professionals because of their intuitive appeal. These relations have so far not been subjected to any formal, systematic study. This is partly because Stevens et al. considered cohesion to be an attribute of design [4]. Thus, they and other subsequent authors have been content with assessing the cohesion of a module by analyzing the natural language description of its purpose [14]. As a result, it has been accepted that "there is no obvious 'measurement procedure' for determining the level of cohesion in a module" [6, page 200].

A question that has not been asked earlier is: Can Stevens et al.'s rules be used to determine the cohesion of a module *from its source code*? Or more accurately: Is there significant agreement between multiple programmers' assessment of the cohesion of modules arrived at by analyzing the source code? This paper presents the results of a controlled experiment conducted to answer this question.

The above questions are important since Stevens et al.'s cohesion relations are paraphrased in several, if not most, software engineering textbooks [10, 12, 20, 21]. There have also been proposals for source code based measures for cohesion [1, 3, 5, 11, 15]. These proposals implicitly or explicitly map their measures of cohesion to the cohesion levels proposed by Stevens et al., although such mappings have not been validated.

A positive answer to the questions being investigated would have important ramifications towards the development and validation of source code based measures for cohesion. It would imply that the claim of mapping between a proposed measure of cohesion and Stevens et al.'s levels may be validated experimentally by comparing the levels (or numbers) assigned by the measure to a module with the cohesion levels assigned to the module by subject programmers. A negative answer, on the other hand, may well suggest the need for further development of the concept of a scale of cohesion. That the cohesion levels assigned to a module by two different programmers may not agree (statistically) will also have another significantly stronger implication. It would raise concerns about the precision of at least one concept being taught in software engineering, an attribute that is clearly not desirable for a discipline to be qualified as "engineering." Such a result will further amplify the need for experimental evaluation of software engineering concepts and principles [8, 7].

The rest of the paper is organized as follows. Section 2 summarizes the notion of cohesion and method for assessing the cohesion of a module as proposed by Stevens et al. Section 3 presents the design of our experiment. Section 4 tabulates the data collected, presents our experimental hypotheses,

Table 1 Associative principle between two processing elements and the corresponding cohesion in increasing order of levels. The cohesion of a module is defined as the lowest cohesion between all pairs of its processing elements.

Cohesion	*Associative principles*
Coincidental	None of the following associations hold.
Logical	At each invocation, one of them is executed.
Temporal	Both are executed within the same limited period of time during the execution of the system.
Procedural	Both are elements of some iteration or decision operation.
Communica-tional	Both reference the same input data set and/or produce the same output data set.
Sequential	The output of one serves as input for the other.
Functional	Both contribute to a single specific function.

and analyzes the data to see if the null hypothesis can be rejected. Section 5 contains orthogonal data that may be used to test the validity of experimental data and/or further support its conclusions. Section 6 concludes the paper by summarizing the result, its implication, and the limitations of our experiment.

2 Stevens et al.'s cohesion classification

In early 70s Constantine attempted to learn why designers associated things into modules. He found that they used certain relationships between a set of actions to determine whether or not they should be performed by the same module. He termed these relationships *associative principles*—principles (properties or characteristics) used by designers to associate actions to be placed in a module. He classified three such principles and arranged them in a linear order (or levels) reflecting the preference of most designers for one principle over another. The ordinal scale defined by the set of associative principles along with their order he termed *cohesion*. Stevens, Myers, Yourdon, and Constantine expanded the list of associative principles to seven [22, 24], which has now become the *de facto* standard and is paraphrased in most software engineering textbooks [10, 12, 20, 21].

Table 1 enumerates the associative principles identified by Stevens et al. and their corresponding cohesion levels. The associative principles give the cohesion between pairs of processing elements. The cohesion of a module on the whole, as per Stevens et al., is defined as the lowest of the cohesion between all pairs of processing elements. The informal definition of processing elements and the associative principles make the concept of cohesion subjective.

3 Experiment design

This section presents the design of our experiment conducted to assess whether Stevens et al.'s cohesion relations are objective. This section describes the background of the experimental subjects, the programs used for the analysis, material (such as questionnaires, etc.) used in the experiment, and the procedure followed. The precise null hypothesis is stated in the next section.

Table 2 Subjects' background information

Subject Background Variable	Low	Mean	High
Years of programming	2	4.6	11
Number of computer science courses in BS	0	10	24
Number of computer science courses in Grad school	3	8	15
Number of programming languages known	1	4	7

Table 3 Subjects' familiarity with the C language and cohesion concepts, on the scale of 0 to 10, where 0: never used/heard, 5: about average, and 10: expert

Subject Background Variable	Low	Mean	High
Familiarity with C language	5	7.6	9
Familiarity with cohesion concepts	3	6.1	8

Table 4 Programs used in the Experiment 1 and their sizes measures

Program Code	Program Name	No. of Lines	No. of Functions	Avg. lines/function
P-1	Expression evaluation	83	5	16.6
P-2	Tax form	161	6	26.8
P-3	Accounting	245	6	40.8
P-4	Bank promotion	172	4	43.0

3.1 Subjects The experiment was conducted as part of a graduate level course on software engineering at the University of Southwestern Louisiana. Fifteen students from this course volunteered to participate, one refrained. Tables 2 and 3 summarize information about their educational background and relevant programming experience. This information was collected in the beginning of the experiment, as discussed later.

3.2 Experimental programs We chose four C programs of a collection of 26 used earlier by Goradia in experiments conducted for his Ph.D. research [9], and made publicly accessible over the Internet. The four programs were selected based upon their modest size and domain of application. The last criterion was used to exclude programs, such as matrix inversion, that required the knowledge of specialized algorithms. Only programs whose domain of application and/or algorithm were expected to be known to Computer Science graduate students, either by virtue of their training or social experiences, were selected.

Table 4 lists the programs chosen for our experiment and presents the size related characteristics of these programs. (The lines of code measure used in this paper corresponds to the number of newline characters.)

3.3 Experiment material The materials used for the experiment may be classified in two categories: (i) material to the administrator of the experiment, (ii) material to the subjects of the experiment. (Interested readers may obtain these documents from the first author.)

The material for the administrator of the experiment contained the following items: (i) description of the experiment, (ii) instructions on how to conduct the experiment, (iii) informed consent forms used to obtain the consent of the subjects, (iv) a copy of the text book section on module cohesion used for the lecture on module cohesion, (v) experiment packets to be given to the subjects, (vi) copies of a quiz to assess the knowledge of the subjects, and (vii) slips containing program and subject codes used for assignment of programs to subjects in a random manner.

The packet given to each subject of the experiment contained the following materials: (i) description of the experiment, (ii) instructions to the subjects, (iii) section on module cohesion from a text book [19], (iv) source code listing of the experimental program assigned to the subject, (v) a data sheet to record the subject's cohesion level assignment; (vi) a remark form to collect the subject's comments about the experiment, and (vii) a questionnaire to collect the subject's educational background.

3.4 Experiment procedure The experiment was conducted in four stages. The activities of each stage are briefly discussed below.

Stage 1: During a regular lecture of the course in which the experiment was conducted, the instructor (Lakhotia, one of the authors) told the students about the experiment that was being planned and its significance to the software engineering community. They were told that subjects were being sought for the experiment, and that the participation was voluntary, and that they were encouraged to participate. It was emphasized that the experiment was not intended to evaluate the participants but was actually attempting to evaluate the material that would be presented to them, and that their participation or non-participation would not influence their grades.

In a lecture about two weeks later, the administrator of the experiment (another volunteer graduate student) once again talked about the experiment and sought volunteers by distributing informed consent forms. Fifteen students volunteered as subjects and signed the informed consent form; one student refrained.

The experiment material was designed so that the names of the participants appeared *only* on the informed consent forms. All other material presented later was coded using subject codes (as described later); the correspondence between a subject and his/her code was never recorded. The students were informed of this scheme to gain their confidence that their privacy would indeed be maintained.

Stage 2: The administrator of the experiment (same volunteer as above) presented a lecture on the concepts of module cohesion. The lecture was based on Stevens et al.'s work. Experiment packets, containing material mentioned earlier, were distributed to the subjects.

The experiment packets were prepared in the following manner. There were a total of sixteen packets, four for every program used in the experiment. Each packet was labeled with a program code and a subject code. The program codes were P-1, P-2, P-3, or P-4 (see Table 4).

Randomness in assignment of subjects to programs was achieved by having the subjects draw from a box the program code they were to analyze. The subjects were assigned code based on their program code (by coding their experiment packet). The subject codes were formed as follows: Sij indicating the $j\mathrm{th}$ subject assigned to the program P_i. Each subject looked at only one program. To ensure anonymity

Table 5 Cohesion assignments for the Expression Evaluation program (P-1)

Function Name	LOC	S-11	S-12	S-13	S-14
compute	23	logical	logical	communicational	logical
operand_value	4	functional	functional	functional	functional
get _token	7	sequential	sequential	functional	procedural
evalute	22	functional	logical	procedural	functional
main	11	communicational	sequential	functional	procedural

Table 6 Cohesion assignments for the Tax Form program (P-2)

Function Name	LOC	S-21	S-22	S-23
initialize	19	temporal	temporal	temporal
schedule_A	15	functional	functional	functional
figure_tax	33	functional	logical	functional
compute_tax	20	sequential	functional	sequential
valid_data	17	temporal	logical	functional
main	26	sequential	functional	sequential

of the subjects, we did not associate subject names with their codes. A maximum of four subjects was assigned to any program.

The subjects took the experiment packets home and studied the program assigned to them. They were given one week to complete their task. They were asked to complete and return items (v), (vi), and (vii) of their packet (listed earlier).

Stage 3: In the third stage, each subject studied the program assigned to him/her and assigned a cohesion level to each function based on the original definition of module cohesion by Stevens et al. The subjects recorded this information on the data sheet provided to them. They also noted their comments about their task on the remark forms and completed a questionnaire about their educational background.

Stage 4: In the fourth stage, the subjects brought to a regular class lecture the completed forms in envelopes provided to them. This lecture too was conducted by the experiment administrator. Before turning in their packets, the subjects were given a quiz on module cohesion. This quiz was to be used to detect and eliminate data by those participants who did not demonstrate an understanding of the basic concepts of module cohesion. The intent of the quiz was not to judge or evaluate the individuals participating in the experiment, but rather to check the validity of the data obtained from the experiment. The subjects enclosed their completed quiz in the packet along with the material completed in Stage 3. These packets were then returned to the administrator.

4 Data analysis and results

4.1 Subjects' Responses The cohesion levels assigned by the experimental subjects to the functions of Programs P-1 through P-4 are presented in Tables 5 through 8. The columns labeled *LOC* contain the size of a function as measured in lines of code.

Table 7 Cohesion assignments for the Accounting program (P-3)

Function Name	LOC	S-31	S-32	S-33	S-34
initialize	15	functional	temporal	coincidental	temporal
change_monthly	22	functional	functional	temporal	sequential
process_ transaction	15	communicational	logical	sequential	communicational
process_ end_ of _month	22	functional	temporal	functional	temporal
process_ report	29	sequential	logical	temporal	procedural
main	41	communicational	logical	coincidental	procedural

Function Name	LOC	S-41	S-42	S-43	S-44
assess_cashflow	31	communicational	logical	communicational	communicational
assess _account _status	19	functional	functional	functional	functional
recommended_account	22	functional	functional	functional	functional
main	57	procedural	sequential	functional	communicational

Table 8 Cohesion assignments for the Bank Promotion program (P-4)

4.2 Hypothesis This experiment investigates whether Stevens et al.'s definition of module cohesion is objective by asking the question: Do the subjects agree on the cohesion of a function in a program? This question may be stated in terms of the following experimental hypotheses:

Null hypothesis(H_0): Subjects are incapable of using Stevens et al.'s relations, and therefore their assignments of cohesion levels will be randomly distributed.

Alternative hypothesis (H_1): Subjects display an above-chance consistency amongst themselves in assigning cohesion levels after learning about Stevens et al.'s relations.

The above null hypothesis is tested using three statistical tests. First we use a nominal statistical test that views the seven levels of cohesion as simple categories, not as scalar values. Next we use analysis of variance (ANOVA) tests that treats the seven cohesion relations to be ordered, as proposed by Stevens et al. Next we perform an omnibus analysis of variance that compares the cohesion levels assigned by the subjects with that assigned by a tool.

4.3 Analysis using nominal statistic We now present the results of analyzing the data using the *binomial* test, a nominal statistical test. For an experiment to qualify as a *binomial* experiment, it must have the following four properties [13]: (a) there must be a fixed number of trials, (b) each trial must result in a "success" or "failure", i.e., it is a binomial trial, (c) all trials must have identical probabilities of success, and (d) the trial must be independent of each other.

These properties are satisfied by our experiment, as follows. (a) For each program used in the experiment, the number of functions when viewed as number of trials is fixed. (b) Each test for agreement on cohesion level assigned to a function results in a success/match or a failure/no-match. (c) All trials have identical probability of success, i.e., the probability of successful match on cohesion level of one function does not affect the probability of successful match on cohesion level of another function. (d) Assignment of cohesion level to one function is independent of the assignment of cohesion level to another function. Thus, the trials are independent of each other.

Table 9 Percentage of agreement amongst subjects for the Expression Evaluation program (P-1)

Program: Expression Evaluation (P-1)			
	S-12	**S-13**	**S-14**
S-11	60%	20%	60%
S-12		20%	40%
S-13			20%

Table 10 Percentage of agreement amongst subjects for the Tax Form program (P-2)

Program: Tax Form (P-2)		
	S-22	**S-23**
S-21	33%	83%
S-22		33%

Table 11 Percentage of agreement amongst subjects for the Accounting Program (P-3)

Program: Accounting (P-3)			
	S-32	**S-33**	**S-34**
S-31	17%	17%	17%
S-32		0%	33%
S-33			0%

Table 12 Percentage of agreement amongst subjects for the Bank Promotion program (P-4)

Program: Bank Promotion (P-4)			
	S-42	**S-43**	**S-44**
S-41	50%	75%	75%
S-42		50%	50%
S-43			75%

Tables 9 through 12 provide, for each experimental program, the percentage of agreement between each pair of subjects.

Assuming that each cohesion level is equally probable, the probability of assigning any particular level of cohesion, i.e., the probability of success, is 1/7 or 0.143. The cumulative probabilities *(p-value)* of success of a binomial test resulting from this probability of success are given in Tables 13 through 16.

An entry with '**' in these tables represents an agreement with a 0.05 level of significance; an entry with a '*' represents an agreement with a 0.10 level of significance; other entries are not statistically significant. The significance level, denoted by α, of a test is the probability of rejecting the null hypothesis when it is true (Type I error in a statistical test) [16].

Table 13 Cumulative binomial probabilities *(p-values)* for the Expression Evaluation program (P-1)

Expression Evaluation (P-1)			
	S-12	**S-13**	**S-14**
S-11	0.023	0.538	0.023
S-12		0.538	0.152
S-13			0.538
S-14			

Table 14 Cumulative binomial probabilities *(p-values)* for the Tax Form program (P-2)

Program: Tax Form (P-2)		
	S-22	**S-23**
S-21	0.207	0.000
S-22		0.207

Table 15 Cumulative binomial probabilities *(p-values)* for the Accounting program (P-3)

Program: Accounting (P-3)			
	S-32	**S-33**	**S-34**
S-31	0.604	0.604	0.604
S-32		1.000	0.207
S-33			1.000

Table 16 Cumulative binomial probabilities *(p-values)* for the Bank Promotion program (P-4)

Program: Bank Promotion (P-4)			
	S-42	**S-43**	**S-44**
S-41	0.101	0.010**	0.010**
S-42		0.101*	0.101*
S-43			0.010**

We can observe from Table 13 through 16 that there is little agreement amongst subjects on the assignment of cohesion to various functions. The only exception to the above observations is the case of program *P-4*. For this program, there is a strong agreement among the subjects *S-41*, *S-43*, and *S-44* with a 0.05 level of significance. As may be seen from Tables 9 and 12, these subjects generally agreed on three of the four functions.

There is no statistically significant evidence to reject the null hypothesis. We also cannot completely reject the alternative hypothesis as the analysis of Bank Promotion program (P-4) supports the alternative hypothesis with a minimum α of 0.1. The results of the test of the hypothesis are therefore inconclusive.

Table 17 Analysis of variance for the Expression Evaluation program (P-1)

Source of variation	SS	df	MS	F
Between functions	38.7	4	9.675	4.21**
Within functions	34.5	15	2.300	
Between subjects	6.0	3	2.000	
Residual	28.5	12	2.375	

Table 18 Analysis of variance for the Tax Form program (P-2)

Source of variation	SS	df	MS	F
Between functions	36.0	5	7.20	2.7*
Within functions	32.0	12	2.67	
Between subjects	5.33	2	2.67	
Residual	26.67	10	2.67	

Table 19 Analysis of variance for the Accounting program (P-3)

Source of variation	SS	df	MS	F
Between functions	21.0	5	4.20	1.03
Within functions	73.5	18	4.08	
Between subjects	32.5	3	10.83	
Residual	41.0	15	2.73	

4.4 Analysis using interval statistical tests In using binomial test, we have analyzed the data for a strict match on categories of cohesion. This test was inconclusive. An alternative to categorical testing is to investigate the extent to which subjects agree that some functions are more cohesive than others, i.e., relative ordering of functions on cohesiveness. For example, we can take subjects' assignment of cohesion levels for a given program as representing values on a scale of cohesiveness, and check the extent to which a group of subjects reliably use such a scale. This may be analyzed using an analysis of variance to examine the reliability of measurements [23, pages 283–289].

Tables 17 through 20 show the results of the analysis of variance test performed after transforming the subjects' cohesion level assignments into numbers ranging from 1 to 7, where 1 represents functional and 7 represents coincidental.

for each of the four programs used in the experiment. An entry for value of F marked with '***' represents reliability at the 0.01 level of significance; an entry with a '**' represents reliability at the 0.05 level of significance; an entry with '*' represents 0.10 level of significance; and other entries are not significant.

Based on ANOVA, the subjects' exhibit some significant agreement on a scale of cohesion for programs P-1 and P-4 at 0.05 or above level of significance, and for program P-2 at 0.1 level of significance. Thus we may reliably reject the null hypothesis that cohesion level assignments are random.

Table 20 Analysis of variance for the Bank Promotion program (P-4)

Source of variation	SS	df	MS	F
Between functions	21.188	3	7.063	7.21***
Within functions	11.751	12	0.98	
Between subjects	2.188	3	0.729	
Residual	9.563	9	1.063	

Table 21 Theta and estimate of the reliability of the mean of the *k* subjects for experimental

Program	Program Code	θ	r_k
Expression Evaluation	P-1	0.6614	0.7257
Tax Form	P-2	0.4157	0.555
Accounting	P-3	-0.0212	-0.0929
Bank Promotion	P-4	1.251	0.833

Table 22 Mean rating for each level of cohesion

Level	1	2	3	7
Mean rating	1.07	1.50	1.73	1.10

Table 21 shows the unbiased theta (θ) and the unbiased estimate of the reliability of the mean of the *k* subjects, r_k, values for each of the programs used in the experiment. These estimates are conservative, as the analysis of variance did not correct for end anchor effects. However, corrected analyses revealed essentially the same patterns of results.

It is worth noting that the analysis of variance reported above are conservative in that they are equivalent to analyses in which subjects are nested within functions. Analyses that take advantage of the potential systematicity in error variance contributed by a given subject across functions display the same essential pattern of results, however. (In these latter analyses the residual mean square is used as the error term.)

Finally, we address the question of the validity of subjects' cohesion assignments. The fact that subjects display systematicity in their ratings need not indicate that they are employing the same ordering as Stevens et al. To address this question, each function was also assigned a cohesion level using an automated tool developed by the first author [15] based on an encoding of Stevens et al.'s rule proposed earlier [11]. The tool found only four levels of cohesion in the subject programs; functional, sequential, communicational, and coincidental. To determine whether the assignments of cohesion levels our subjects gave corresponded to those of Stevens et al., the levels assigned by the tool were treated as constituting the independent variable in an analysis of variance examining rated cohesion. A single rating of cohesion for each cohesion level rated by a subject was obtained by averaging that subject's rating for all functions at that level. The mean rating for each level constituted the dependent data in the analysis. The resulting data were then subjected to an omnibus analysis of variance in which subjects were treated as being nested within cohesion level.

Table 23 Omnibus analysis of variance of the mean ratings

Source of variation	Sum of Squares	df	Mean Square	F
Between	12.169663	3	4.056554	2.42
Within	43.7	26	1.674244	

In this analysis, eleven subjects contributed ratings for cohesion level 1 (functional); eight contributed ratings for cohesion level 2 (sequential); four contributed ratings for cohesion level 3 (communicational); and seven contributed to cohesion level 7 (coincidental).

The mean rated cohesiveness for these four levels is presented in Table 22, and the analysis of variance on these data is summarized in Table 23. The analysis revealed a significant difference in ratings as a function of cohesiveness at the 0.10 level. As may be seen from Table 22, the data do exhibit an orderly increase in rated level with actual increases in level with the exception of the coincidental level. Thus, while our subjects do not assign cohesion in a fashion perfectly consistent with Stevens et al., there is some evidence that they exhibit some of the same sensitivity to cohesion as found with Stevens et al. scale. At the same time, the actual mean ratings suggest that our subjects do not adopt the precise categories of the Stevens et al. scale, accounting for the mixed results obtained with nominal statistics.

The results suggest that the notion of levels of cohesion is one that subjects may use with some reliability and validity, although more work needs to be done exploring why functional and coincidental code was given equivalent ratings.

4.5 Power of the experiment The power of a statistical test is defined as the probability of rejecting H_0 when H_0 is false. This probability is equal to $1 - \beta$, where β is the probability of a Type II error. A Type II error occurs when we fail to reject H_0 when H_0 is false. It is not possible to specify the exact value for the power of statistical tests conducted in this experiment because H_1 is an inexact hypothesis. Given the approach adopted here, this was not of grave concern. Nevertheless, some factors that affect the power of statistical tests in our experiment are outlined here:

- The power of a statistical test can be increased by avoiding Type II errors. If Type II errors are to be avoided, then a relatively large sample size or a large α value is required. As the sample size increases, the power of a statistical test increases. In our experiment, since the sample size is too small (at most four subjects per program), the power of the experiment is weak. As the alpha level becomes more stringent (goes from .05 to .01), the power decreases. This situation did not happen in our experiment as we have used alpha level of 0.05 or 0.10. In our experiment, the major contributor to the weak power seems to be the small sample size.
- In our experiment, the programs used did not contain many functions (at most six functions per program) and majority of these functions displayed functional cohesion, as measured by the tool. Therefore, the stimulus materials used did not contain sufficiently large number of functions and the functions did not exhibit a good distribution of cohesion levels. These conditions also affected the power of the experiment.

The power of such an experiment could be increased by assigning more subjects per program, by employing *within-subjects* design (as opposed to *between subjects* design), by using programs with sufficiently large number of functions and/or by using programs whose functions exhibit a good distribution of cohesion levels.

5 Analysis of subjects knowledge of cohesion

In the last stage of the experiment, each subject had completed a quiz and a questionnaire. These provide a means to evaluate the quality of the data collected in the experiment. This section summarizes and analyzes the resulting feedback.

5.1 Subjects' performance on quiz The quiz was designed to gauge whether the subjects had a minimal understanding of the concept of cohesion. Using Bloom's taxonomy of educational objectives, the minimal level of knowledge of a subject in an area constitutes knowing its terminology and facts [2]. The quiz was designed to test this minimal knowledge because this is sufficient to assess whether the students have grasped the description of each relation, as given by Stevens et al.

The quiz contained the following types of questions (a) one true/false question regarding the definition of module cohesion, (b) one question in which a list of cohesion levels in no particular order had to be rearranged in increasing order of cohesion, (b) seven fill-in the blank type questions where the definition of a cohesion level had to be related to the name of the cohesion level, and (d) six small code fragments, commonly used in the literature on module cohesion, for which the subjects had to assign a cohesion level.

The subjects' answers in the first three components of the quiz were 95% correct which shows that they understood the concepts of module cohesion. Their answers in the last component (assignment of cohesion to code fragments) had some variations. This again shows that the original definitions are subjective in nature and difficult to apply to determine precisely the cohesion of a code fragment.

5.2 Feedback from the subjects The subjects were also given a form to provide remarks about the experiment. The form contained a set of questions, given below, and space to provide descriptive feedback.

1) How did you find the lecture on module cohesion to understand?
2) How did you find the material on module cohesion to read and understand?
3) How did you find the definitions of cohesion levels to understand?
4) How did you find the program assigned to you to understand?
5) How did you find the task of assigning of cohesion level to functions?

Each of these questions was answered using one of the following three responses: *easy*, *average*, *difficult*, or *very difficult*.

The responses to the above questions are summarized in Table 24. Each cell in this table gives a count. For each question, the table gives the distribution of the subjects' responses in the four levels of difficulty: easy, average, difficult, and very difficult.

Notice that none of the subjects felt that the material on module cohesion was difficult to understand (Q. 2); 86.6% felt that the definitions of cohesions were not difficult to understand, i.e., were either easy or average (Q. 2); 93.3% found the programs said that the sample programs were not difficult; yet only 73.3% reported that the task of assigning cohesion level to functions (Q. 5) was not difficult.

Table 24 Summary of feedback information from subjects. Each row gives the distribution of responses across various levels of difficulty.

Question #	Easy	Average	Difficult	Very difficult
1.	4	7	3	1
2.	8	7		
3.	8	5	2	
4.	10	4	1	
5.	3	8	4	

These responses would indicate that by and large the students understood the programs and the cohesion material well. Yet there is a significant variation in their assignment of cohesion levels to the subject program. One reason for this variation may be that the original definitions of module cohesion by Stevens et al. are very intuitive, hence are easy to understand. Since they are not objective, they can be interpreted differently leading to the significant variation.

This analysis is supported by subjects' descriptive comments, some of which are quoted below:

1. "The definition of levels of cohesion are intuitive. Therefore, assigning a level of cohesion to a function is also intuitive. It is hard to do objectively."
2. "The first three levels of cohesion (functional, sequential, communicational) were straightforward and easy to understand. The remaining levels of cohesion were confusing." (This is consistent with the results of omnibus analysis reported above.)
3. "It is not very easy to give an accurate cohesion level assignment for the given source code."
4. "Making the final choice of cohesion level for each function is only as good as my best estimate."

6 Conclusions

This paper reports the result of a preliminary study conducted to assess whether Stevens et al.'s rules of cohesion are objective. The question is significant since these rules are paraphrased in most software engineering textbooks [10, 12, 20, 21]. A study of this nature would help evaluate the precision of material taught in software engineering curriculum. Our own interest in this study was kindled due to our interest in developing a measure for cohesion by simply translating into logic the rules of cohesion proposed by Stevens et al.[11, 15]. Since such a translation, to the extent possible, preserves the intent of the original rules, our measure could be treated as a reference measure for comparing other proposed measures [1, 3, 5, 17, 18]. This experiment is a step in that direction.

In our experiment fifteen subjects (graduate student programmers) were trained on the concepts of cohesion and were then asked to classify stimulus materials (four programs). The data collected was subjected to various analyses. A nominal statistical test for a precise match between the levels assigned by the programmers could not reject the null hypothesis that programmers assigned cohesion levels at random. This test was inconclusive in that it did not reject the alternative hypothesis either, i.e., the subjects were mutually consistent in their assignment of cohesion levels. An analysis of variance examining the reliability of the measurements, however, indicated that the subjects exhibit some significant agreement. This implies that there is a significant agreement in the relative ordering of functions created by virtue of the cohesion levels that the subjects assigned. This correspondence of the relative order

of levels was further tested against Stevens et al. scale by analyzing the mean rating for each level of cohesion using omnibus analysis of variance. This analysis indicated some degree of validity in the subjects ordering of the first three levels of cohesion.

The above findings were further supported by an orthogonal set of data resulting from a quiz and a questionnaire that our subjects completed at the end of the experiment. This data indicates that although the students showed good grasp of the definition of various cohesion levels, they reported difficulty in applying the cohesion levels (even though the programs were not reported to be difficult). This difficulty was also reflected by the significant variation in the subjects' performance on textbook type exercises for assigning cohesion levels contained in the quiz. In their descriptive comments the subjects expressed comfort with assigning the first three levels, further supporting the results of the omnibus analysis.

The experiment lacked sufficient power to be conclusive, nonetheless it does provide us with data to draw some preliminary conclusions. We first take a closer look at the Stevens et al.'s notion of cohesion and then relate it to the above results.

The Stevens et al. scale has two components: (a) an ordered set of seven levels and (b) a set of properties to be satisfied by components placed in each level. The levels are ordered based on the desirability of the corresponding properties, where less desirable properties are related to lower levels. Hence, the level assigned to a component is commonly taken as a reflection of its "quality." (Note that the specific quality attribute that a cohesion level is an indicator of has not yet been studied, and hence we leave it undefined.) The ordering of the levels gives the impression that it is an *ordinal* scale. However, since each level has a "precise" property associated to it, a component can only be assigned a level if it satisfies that property. Since there is no obvious continuity or fuzzy boundaries in the properties, two programmers should essentially place a given component in the same level. This implies that Stevens et al.'s scale is really *absolute* (an argument that has not been earlier made in the literature).

Our test for whether there is an "absolute" agreement in the cohesion level assigned by programmers was inconclusive, whereas we find that there is a significant agreement in the relative order in which the components are placed (by virtue of the cohesion levels assigned to them). One possible explanation for this may be that when the programmers found it difficult to assign precise cohesion levels based on Stevens et al.'s properties, they instead assigned cohesion levels such that the components were ordered according to their perception of the components' quality. This would explain the significant agreement in the relative order of functions resulting from the cohesion assignment. This explanation could be validated by having a control group of programmers simply order program components on the basis of their perceived quality. A confirmation of the reliability of the programmers measurement would indicate that such experiments can indeed be used to validate software measures.

The variations in the relative ordering of components, as indicated by the difference between a level and its mean rank, could be because some component pairs may be much more difficult to order. This is analogous to the variations that may be found in response to the question: Place the following animals in the decreasing order of their average weights – horse, duck, rooster, mouse. For our program components such variations would not be expected if the properties associated with each level were precise and easy to determine. That the programmers found the stimulus programs and the notion of cohesion to be easy to understand reduces the possibility that the variations were due to some inherent difficulty in the programs or the concept. It could possibly be explained by the descriptive feedback of the programmers that the concepts were hard to apply.

The above preliminary findings that programmers find it difficulty to apply cohesion rules are important since they raise questions about the precision of at least one concept taught in software engineering. Our finding is important in the light of growing concerns about lack of experimental evidence in support of software engineering principles, and further substantiates the need for such experimental investigations [8, 7].

Acknowledgments: The authors wish to thank Anurag Bhatnagar and John Gravley for their help in conducting the experiments; Robert McFatter for his advise on experimental analysis; and the fifteen students whose participation made this study possible. This work was partially supported by grants from the Army Research Office (ARO DAAH04–94–G-0334) and the Louisiana Board of Regents (LEQSF 1991-92 ENH-98 and 1993–95 RD-A-38).

Bibliography

[1] James M. Bieman and Linda M. Ott. Measuring functional cohesion. *IEEE Transactions on Software Engineering*, 20(8):644–657, August 1994.

[2] B. Bloom. *Taxonomy of Educational Objectives: Handbook I: Cognitive Domain*. David McKay, New York, 1956.

[3] Germinal Boloix and Pierre N. Robillard. Interconnectivity metric for software complexity. *INFOR*, 26(1), 1988.

[4] Lionel Briand, Sandro Morasca, and Victor R. Basili. Defining and validating high-level design metrics. Technical Report CS-TR-3301, University of Maryland, Baltimore, Maryland, 1994.

[5] Thomas J. Emerson. A discriminant metric for module cohesion. In *Proceedings of the 7th International Conference on Software Engineering*, March 1984.

[6] N.E. Fenton. *Software Metrics: A Rigorous Approach*. Chapman & Hall, Van Nostrand Reinhold, 1991.

[7] Norman Fenton, Shari Lawrence Pfleeger, and Robert L. Glass. Science and substance: a challenge to software engineers. *IEEE Software*, 11(4):86–95, July 1994.

[8] Robert L. Glass. The software research crisis. *IEEE Software*, 11(6):42–47, November 1994.

[9] T. S. Goradia. *Dynamic impact analysis: analyzing error propagation in program executions*. PhD thesis, Dept. of Computer Science, New York University, November 1993.

[10] Pankaj Jalote. *An Integrated Approach to Software Engineering*. Spring-Verlag, 1990.

[11] Arun Lakhotia. Rule-based approach to computing module cohesion. In *Proceedings of 15th International Conference on Software Engineering*, pages 35–44, May 1993.

[12] A. Macro and J. Buxton. *The Craft of Software Engineering*. Addison-Wesley, 1987.

[13] F. Mosteller, R. E. K. Rourke, and G. B. Thomas. *Probability and Statistics*. Addison-Wesley Publishing Company, 1967.

[14] Glenford J. Myers. *Composite/Structured Design*. Van Nostrand Reinhold Company, 1978.

[15] Jagadeesh Nandigam. *A measure for module cohesion*. PhD thesis, University of Southwestern Louisiana, 1995.

[16] J. Newmark. *Statistics and probability in modern life*. Sanders College Publishing, a Harcourt Brace Jovanvich College Publisher, 5th edition, 1992.

[17] Linda M. Ott and Jeffrey J. Thuss. The relationship between slices and module cohesion. In *Proceedings of the 12th International Conference on Software Engineering*, May 1989.

[18] Linda M. Ott and Jeffrey J. Thuss. Slice based metrics for estimating cohesion. Technical Report CS-TR-91-04, Michigan Technological University, November 1991.

[19] M. Page-Jones. *The practical guide to structured systems design*. Yourdon Press Computing Series, 2nd edition, 1988.

[20] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw Hill, third edition, 1992.

[21] M. L. Shooman. *Software Engineering*. McGraw-Hill, 1983.

[22] W. P. Stevens, G. J. Myers, and L. L. Constantine. Structured design. *IBM Systems Journal*, 13(2):115–139, 1974.

[23] B. J. Winer. *Statistical principles in experimental design*. McGraw-Hill Book Company, 2nd edition, 1971.

[24] Edward Yourdon and Larry L. Constantine. *Structured Design*. Yourdon Press, 1978.