Technical Report TR-1442 September 1984

Structural Coverage of Functional Testing

Victor R. Basili
James Ramsey

Department of Computer Science
University of Maryland
at College Park

Abstract

A large, commercially developed FORTRAN program was modified to produce structural coverage metrics. The modified program was executed on a set of functionally generated acceptance tests and a large sample of operational usage cases. The resulting structural coverage metrics are combined with fault and error data to evaluate structural coverage in the SEL environment.

We can show that in this environment the functionally generated tests seem to be a good approximation of operational use. The relative proportions of the exercised statement subclasses (executable, assignment, CALL, DO, IF, READ, WRITE) changes as the structural coverage of the program increases. We propose a method for evaluating if two sets of input data exercise a program in a similar manner.

We also provide evidence that implies that in this environment, faults revealed in a procedure are independent of the number of times the procedure is executed and that it may be reasonable to use procedure coverage in software models that use statement coverage. Finally, the evidence suggests that it may be possible to use structural coverage to aid the management of the acceptance test process.

This study is funded by NASA grant NSG-5123.

1. Introduction

The goal of this study has been to understand and improve the acceptance test process in the NASA Goddard Space Flight Center SEL environment [SEL82a]. Towards this end, an SEL program has been modified to produce structural coverage metrics. The instrumented program, the MAL language preprocessor, is a subset of the RADMAS satellite attitude maintenance system [STL82]. It has 68 functions and subroutines, 10k source lines of code and 4k executable statements. The program was modified to measure both procedure coverage and statement coverage. Coverage is also computed for the statement subclasses: assignment statements, CALL, DO, IF, READ, and WRITE.

The modified program was executed on a set of ten functionally generated acceptance tests and on sixty typical operational usage cases [CSC78]. Error, fault and failure data* were collected from the system test through operation phases [SEL82b]. Each execution of an acceptance test or an operational usage case provides a structural coverage statistic. These structural coverage statistics are first examined individually to understand the static properties of the acceptance test process. Randomly generated sequences of acceptance tests and operational usage cases are then examined in an attempt to understand the dynamic properties of structural growth. Finally the coverage data are combined with the error, fault and failure data to understand how faults are revealed.

---

* We have tried to follow the IEEE Standard Glossary of Software Engineering Terminology definitions of error, fault and failure: An error is the "human action that results in software containing a fault." A fault is "a manifestation of an error." A failure is "a departure of program operation from program requirements" [IEEE83]. Some of the sources we cite were written before the standard; their use of error may differ from the standard.

2. Goals of the Study

The first goal of this study was to characterize structural coverage in the SEL environment. The first questions address the simple, static properties of structural coverage for the different kinds of inputs. Question I.D compares two kinds of structural coverage: procedure and statement coverage. It would be a useful result if we could show that procedure coverage can be substituted for statement coverage in software models since procedure coverage is easier to measure than statement coverage. The final question addresses the dynamic properties of structural coverage: the structural coverage growth of a set of input cases.

Some testing strategies ([Duran80] and [Dyer82]) and software reliability models [Brooks80] require a method for showing that two sets of inputs exercise a program in a similar fashion. This motivated goal II: "Can different input sets be differentiated using structural coverage metrics?" Questions II.A-II.D explore several methods of doing this.

The purpose of the functional tests is to reveal faults in the program yet some faults are still revealed in operation. What classes of faults does functional testing miss? Does operational use exercise the code differently than the functional tests? How is this related to structural coverage? This motivated the next goal: "How are faults and structural coverage related?" Questions III.A - III.D analyze the SEL error, fault, and failure data with respect to structural coverage [SEL82b].

Finally in Section IV these ideas are combined to suggest an improved method of managing acceptance tests.

I. Characterize structural coverage in the SEL environment.

I.A. What is the statement coverage of functional testing? What is the procedure coverage of functional testing?

I.B. What is the statement coverage of operational use? What is the procedure coverage of operational use?

I.C. What are the intersection / union of functional testing and operational use?

I.D. Can procedure coverage be substituted for statement coverage in software models?

I.E. What are the properties of structural coverage growth?

II. Differentiate different input sets using their structural coverage.

II.A. Are heavily exercised procedures more likely to contain a fault?

II.B. Using Venn diagram?

II.C. Using nonparametric statistics?

II.D. Using number of executions of prime sections of code?

III. Relate errors, faults, failures and structural coverage.

III.A. Are more heavily exercised procedures more likely to contain a revealed fault?

III.B. Are faults related to time to isolate?

III.C. Are faults related to time to understand and implement?

III.D. Are faults related to type of error?

IV. Use structural coverage to aid the management of acceptance tests.

IV.A. Can structural coverage be used to suggest new acceptance tests?

IV.B. Can structural coverage be used to improve reliability models?

Figure 1.

3. Data and Analysis

This section contains a description of the data and their analysis paralleling the outline in figure 1.

3.1. Structural Coverage in the SEL

Question:

> What is the statement coverage of functional testing? What is the procedure coverage of functional testing?

The acceptance tests we used are functional or "black box" tests [Howden81], [Myers79]. Since exhaustive sampling of the input subdomains is impractical, a few sample inputs from a few subdomains are chosen that the testers feel are likely to reveal faults [CSC78]. There are 17 acceptance tests.

Table 1 shows the structural coverage of the acceptance tests. Test 1 exercised 33 out of 68 possible procedures. It exercised 1069 of the 4300 executable statements. In total the 17 tests exercised 51 procedures and 2408 executable statements (Union). There were 778 executable statements that were exercised by every test case (Intersection). These numbers are interpretated as percentages of the maximum in table 2.

Please note that we did not measure the structural coverage of either system or unit tests. Statements which were not exercised during acceptance test might have been exercised during previous testing. Structural coverage measures were not available during either system or unit test. Procedures were not tested with the goal of achieving high structural coverage.

Question:

What is the statement coverage of operational use? What is the procedure coverage of operational use?

We obtained 60 input cases that we claim are representative of SEL operational usage. These are 60 samples of actual operational usage cases. This is significantly different from other definitions of operational usage where typically the input domain is divided into subdomains, with each subdomain being assigned a probability of execution. Input cases are then chosen using the probabilities of execution [Brown75], [Duran78], and [Dyer82]. Our definition of operational usage lacks both the definition of subdomains and the assignment of probabilities. These probabilities are difficult to compute and verify. Rigorously derived or otherwise, these operational usage cases define how the program was exercised.

The statement and procedure coverage of operational usage is displayed in tables 3-4.

Question:

What are the intersection / union of functional testing and operational use?

Table 5 compares the structural coverage of functionally generated acceptance tests and operational usage. Together they exercised 55 procedures and and 2768 executable statements. Their intersection (the statements exercised by both sets of inputs) contains 51 procedures and 2397 executable statements. There are 360 executable statements that are exercised by operational usage but not by acceptance test; 11 statements that are exercised by acceptance test but not by operational usage. Table 6 shows the raw numbers interpreted as percentages.

Some interesting observations can be made. The I/O statements, especially the WRITE statements, are less likely to be executed than most other statement subclasses. This is reasonable considering the role WRITE statements play in debugging and error condition handling code. Also, as statement coverage increases, different statements subclasses are more likely to be exercised. In table 6 the line labeled "OpU-A" describes the statements that are executed in operational use but not in acceptance test. Operational usage exercised 8.4% of the code that acceptance test never exercised. This 8.4% is not an even cross section of the statement subclasses. One would reasonably expect the 8.4% to be similar for different statement subclasses but this is not so. 12.1% of the IF statements are executed; half again as much as might be expected.

While this is an interesting result in its own right, this also has some significance to software reliability models. Assuming that statements from different statement subclasses have different likelyhoods of being a "fault," then this result seems to imply that a representative reliability model should have a hazard function (see [Myers79]) that varies over time.

Question:

Can procedure coverage be substituted for statement coverage in software models?

Statement coverage is easy to measure but it is costly in terms of execution time; procedure coverage is much cheaper to measure. Showing that procedure coverage could be substituted for statement coverage in software models would be a useful result. We have not tried to substitute procedure coverage for statement coverage in software models to demonstrate our hypothesis, but rather we have discovered a result that seems to support this possibility.

Each execution of the instrumented program produced both a procedure coverage statistic and seven statement coverage statistics (the number of assignment statements, executable statements, CALLs, DOs, IFs, READs, and WRITEs exercised). Plots 1-14 show procedure coverage versus the different kinds of statement coverage for both operational usage and acceptance test. The plots seem to be linear. This is unremarkable; the more procedures that are exercised, the more statements are exercised. What is interesting is the tightness of the linear fit. In the limited range we examined, the values are never more than ±200 statements from the estimate for acceptance test and ±300 statements for the operational usage cases.

Question:

What are the properties of structural coverage growth?

For a set of input cases, structural coverage monotonically increases with the execution of each new input case (bound above by the number of reachable statements). This section examines the growth of structural coverage. It is important for two reasons:

(1) It provides a way to see if two sets of input cases exercise the program the same way. This provides a way to compare the

equivalence of operational use and acceptance testing.

(2) It provides useful data for the reliability models. Assuming that increased coverage implies a higher failure rate, then anything we learn about the growth of structural coverage can be applied to the calculation of the reliability models' hazard functions.

With 17 acceptance tests and 60 operational usage cases, there are clearly too many sequences to exhaustively examine. In a personal communication, Amrit Goel proposed a solution: examine the structural coverage of a large, but manageable number of sequences. Plots 15-16 show the structural coverage growth of 100 permutations of both acceptance tests and operational usage, with median and quartiles superimposed. Note that the acceptance test plots must all end at the point (17, 2408).

A variety of models were fitted to the structural coverage growth data in an attempt to learn more about structural coverage growth. A good mathematical model of structural coverage growth would provide insight into structural growth. Models were fitted to the first half of a sequence to evaluate their usefulness as predictors and to the entire sequence to evaluate their ability to characterize structural coverage growth. Plots of the residuals were examined visually to estimate goodness of fit.

The best fit was obtained using Goel and Okumoto's NHPP model [Goel80b]. The NHPP model was originally defined as a reliability model. Given a history of faults revealed over time, it predicts the number of faults revealed by time t. It is being used here as a model of structural coverage growth. Restated in terms of structural coverage growth, the model is:

$$m(t) = a(1-e^{-bt})$$

where m(t) is the number of statements executed after test t. a predicts the maximum number of statements to be executed. b defines the steepness of the curve. Given m(1) through $m(t_{max})$, a and b can be calculated. Note the following properties:

$$m(0) = 0$$
$$m(t_{max}) = SC(t_{max})$$
$$\lim_{t \to \infty} m(t) = a$$
$$= \text{maximum statement coverage}$$

It is the best of the models attempted, but its results are imperfect even when a variety of data transformations are applied. Plots 17-24 show some of the fitted models and their residuals. This remains an area of future research.

In summary, we have used structural coverage to provide insight into how functional acceptance test and operational usage exercise a program's code; to suggest results that effect reliability models; to suggest a relationship between procedure coverage and statement coverage; and to move toward understanding statement coverage growth.

3.2. Comparison of Inputs Using Structural Coverage Metrics

Does functional testing have the same coverage profile as operational usage, or more generally, can structural coverage be used to compare two sets of program inputs? This question is interesting for two reasons:

(1) Some testing models require input sets that are "representative" of operational usage [Brown75]. Structural coverage could provide a way of measuring this.

(2) Many reliability models, when using past failure data to predict failure rate or number of failures, assume that the past inputs are similar to the present inputs. Structural coverage could provide a method for confirming this.

Question:

Can the Venn diagram technique be used to differentiate input sets?

In section 3.1 we compared functional test sets with operational usage using a Venn diagram technique (tables 5-6). We used this to show differences in the way operational usage exercised the program. Could this be extended to other input sets? For example, it seems plausible that tests generated with the goal of high branch coverage would execute different code than tests generated by test mutation on arithmetic expressions [DeMillo78] or that boundary value functional tests would exercise different sections of code than statistical predictions of operational usage. We hypothesize that the code in the different sections of the Venn diagram would reflect the properties of the two sets of tests.

Question:

Can input sets be differentiated using nonparametric tests of structural coverage?

Acceptance test and operational usage were statistically compared using both the Mann-Whitney and Kruskal-Wallis tests*. The proposed hypotheses were: "For each of the structural coverage classes (procedures, executable statements, assignment statements...) the population represented by the 60 operational usage cases is similar to the population represented by the acceptance test cases."

Table 7 shows the Kruskal-Wallis H statistic. It shows the result of the test (reject or fail to reject) and the appropriate significance level for each statement class. Table 8 shows the results for the Mann-Whitney U statistic. The column "low U" shows which population had the lower central tendency.

The tests failed to reject the hypotheses for all statement types except READs. Since there are so few READ statements, a small, random difference in the tests could falsely manipulate the statistic. The other statement classes are less susceptible to small changes and represent a better population to examine.

The tests fail to reject the hypotheses that the two populations are similar, meaning that in this case, operational use and acceptance test cannot be distinguished by their structural coverage numbers.

Question:

Can the number of executions of prime sections of code be used to differentiate input sets?

---

* The Mann-Whitney and Kruskal-Wallis tests were chosen because they are nonparametric tests; they make no assumptions about the distributions of source populations. The Mann-Whitney test is most sensitive to differences in "location (central tendency)." The Kruskal-Wallis test is sensitive to differences in "location or dispersion or skewness." [Siegel56].

Are statements executed as thoroughly by acceptance test as they are by operational usage? For each statement in the program, it is possible to count how many times it was exercised by a particular acceptance test or operational usage case. (This differs from the number of times it was executed). If acceptance test and operational usage are similar, then the percentage of acceptance test cases that executed a statement should be similar to the percentage of operational usage cases.

The two percentages were calculated for each prime section of code. The plotted data are shown in scatter plot 25. The regression line has slope 0.921 and intercept 0.032. The r square value is 0.863.

Since the plot does not show any imbalance, one could conclude that acceptance test and operational usage exercise the code equally thoroughly. It is a future goal of this research to replace this empirical judgement by a statistical test.

To summarize, we proposed three methods for comparing sets of program inputs: Venn diagram comparison of executed statements, statistical comparison, and thoroughness of execution of prime sections code. These methods may be able to differentiate input sets, a result that would be useful for understanding reliability models and some testing strategies.

3.3. Error, Faults, and Failures and Structural Coverage

The SEL has been collecting data on software development for 7 years [SEL82a]. Error, fault and failure data are collected using the "Change Report From" or CRF (see figure 2). A CRF is filed whenever a change, enhancement or fault repair is made to a subroutine or data file. This study examines the fields "time to isolate the error," "the time to understand and implement," and the section "type of error*."

There were eight faults found during operation. Each fault could be repaired by changing code in one procedure. One procedure contained two faults. With these data, we can address these questions:

Question:

Were heavily exercised sections of code more likely to contain faults?

These data are shown in table 9. A mark is entered for each of the 68 subroutines. The vertical axis describes the number of times the subroutine was exercised in operational usage. The subroutines that contained the faults are marked with "*".

Half of the procedures were exercised by more than 90% of the operational usage cases. About half of the revealed faults occurred in this group of procedures (3 of 8). With these data we reject the hypothesis that more heavily exercised subroutines are more likely to contain a revealed fault.

Tables 10-12 show faults categorized by time to isolate, time to understand, and number of times the procedure was exercised.

Question:

Is procedure coverage related to time to isolate?

---

* Time to isolate the error is classified as taking: less than one hour, one hour to one day, greater than one day, never found. Time to understand and implement the change is classified as taking: less than one hour, one hour to one day, one day to three days, or greater than three days. Faults are categorized as originating in the: requirements, functional specification, design (either involving data or expression), external environment, use of language, clerical or other.

Time to isolate the change seems to be independent of procedure coverage.

Question:

Is procedure coverage related to time to understand and implement?

Increased usage seems to be associated with longer time to understand and implement a change. This might be explained by suggesting that the lightly exercised procedures contain fairly simple code while the heavily exercised code is, by necessity, more complicated and requires more time to modify.

Question:

Is procedure coverage related to type of error?

Table 12 lists the faults classified by type and procedure coverage in operational usage. There are too few faults to reveal any interesting patterns.

In summary, we have tried to relate statement coverage to: "time to isolate an error," "time to understand an error," and "type of error." The data begins to suggest a relationship between "time to understand an error" and structural coverage. There were too few errors to make any firm statements about "time to isolate an error" and "type of error." This remains a promising area of study.

4. Structural Coverage and the Management of Acceptance Tests

Combined with failure data, structural coverage could aid the design of acceptance tests. Imagine a manager in charge of designing acceptance tests for a group of similar projects or for various releases of a single project. With the failure data from the previous project or release and the structural coverage of both the acceptance and operational usage cases he can suggest new acceptance tests for the next release. He could require tests to exercise unexercised sections of code. He could require new acceptance tests to explain the code missed by acceptance test but exercised in operational usage. If he is using a testing methodology or reliability model that requires inputs that are representative of operational usage, he can use these data to select more representative tests.

We see structural coverage being used by a manager in an iterative fashion:

(1) Gather structural coverage data on acceptance tests and release the project.

(2) Gather structural coverage data and failure data on operational usage. Use these data to adjust reliability models.

(3) Use structural coverage data to: suggest new tests and evaluate how the old tests were created.

(4) Restart the cycle with the new acceptance tests.

5. Conclusions and Criticisms

We conclude:

(1) We may be able to compare sets of inputs using statistical tests and Venn diagram techniques. This would be useful for examining some testing methods and reliability models.

(2) The structural coverage growth of different statement subclasses grows at different rates. This insight might be of interest to reliability model developers.

The data seem to imply:

(1) Faults are independent of number of executions. We can (in our environment) reject the hypothesis that heavily exercised procedures are more likely to contain more revealed faults.

(2) Procedure coverage may be used for statement coverage.

(3) Management of the acceptance test process is possible.

This study can be criticized on a number of points:

(1) There are too few faults to make any forceful statements about errors, faults, failures and structural coverage. (But then again we cannot fault NASA/GSFC for having programs with too few faults.)

(2) While the data suggests that it may be possible to differentiate test sets using structural coverage, we have never provided an example that shows that it can!

(3) This study does not address the order in which the functional tests were used, the order of the operational usage cases or which operational usage cases revealed the faults.

(4) The study did not produce a good model of structural coverage growth.

These points will be addressed when the study is replicated in the summer and fall of 1984. The program being studied is DERBY [CSC83], a large (300 routines, 50k source lines of code), satellite simulator.

The new project is larger and should have more faults. With the new project, we will gather more thorough information on the order of system tests, acceptance tests, operational usage cases, plus the exact input that reveals a failure. The results of this new study should answer many of the questions raised by this study.

6. Acknowledgments

We would like to thank Frank McGarry, Dr. Gerald Page, and Dr. Amrit Goel for their help in the production of this paper, Dr. David Hutchens for a clear-eyed review, and the University of Maryland's Software Engineering group for providing a fertile intellectual environment.

7. References

[Basili81]
Basili, Victor R. and David M. Weiss, Evaluation of a software requirements document by analysis of change data, Proceedings of the Fifth International Conference on Software Engineering, San Diego, CA, pp. 314-323, March 9-12, 1981.

[Basili82]
Basili, Victor R. and David M. Weiss, A Methodology for Collecting Valid Software Engineering Data, TR-1235, Computer Science Technical Report Series, December 1982.

[Brooks80]
Brooks, W. D. and R. W. Motley, Analysis of Discrete Software Reliability Models, RADC TR 80-84, RADC, April 1980.

[Brown75]
Brown, J. R. and M. Lipow, Testing for software reliability, Proceedings of the International Conference on Reliable Software, Los Angeles, CA, pp. 518-527, April 1975.

[CSC78]
Acceptance Test Methods, TM-78/6296, Computer Sciences Corporation, October 1978.

[CSC83]
ERBS Dynamics Simulator User's Guide and System Description, SD-83/6044, Computer Sciences Corporation, August 1983.

[DeMillo78]
DeMillo, Richard A., Richard J. Lipton, and Frederick G. Sayward, Hints on test data selection: Help for the practicing programmer, Computer, pp. 34-41, April 1978.

[Duran78]
Duran, Joe W. and John J. Wiorkowski, Towards models for probabilistic program correctness, Proceedings of the ACM Software Quality Assurance Workshop, pp. 39-44, 1978.

[Duran80]
Duran, Joe W. and John J. Wiorkowski, Quantifying software validity by sampling, IEEE Transactions on Reliability **R-29**, 2, pp. 141-144, June 1980.

[Duran81]
Duran, Joe W. and Simon Ntafos, A report on random testing, Proceedings of the Fifth International Conference on Software Engineering, pp. 179-183, March 1981.

[Dyer82]
Dyer, M. and Harlan D. Mills, Developing electronic systems with certifiable reliability, Proceedings of the Conference on Electronic Systems Effectiveness and Life Cycle Costing, NATO Advanced Study Series, Springer-Verlag, Summer 1982.

[Goel80a]
Goel, Amrit L., Software error detection model with applications, Journal of Systems and Software **1**, 3, pp. 243-249, 1980.

[Goel80b]
Goel, Amrit L. and K. Okumoto, A Time Dependent Error Detection Rate Model for Software Performance Assessment with Applications, annual report to RADC, Department of Industrial Engineering and Operations Research, Syracuse University, Syracuse, New York, March 1980.

[Goodenough75]
Goodenough, John B. and Susan L. Gerhart, Toward a theory of test data selection, IEEE Transactions on Software Engineering, pp. 156-173, June 1975.

[Howden81]
Howden, William E., A Survey of Dynamic Analysis Methods, Tutorial: Software Testing & Validation Techniques, 2nd Ed., ed. E. Miller and W. E. Howden, pp. 209-231, 1981.

[IEEE83]
IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 729-1983, IEEE Inc., February 1983.

[Musa80]
Musa, John D., Software reliability measurement, Journal of Systems and Software 1, 3, pp. 223-241, 1980.

[Myers79]
Myers, G. J., The Art of Software Testing, John Wiley & Sons, New York, 1979.

[SEL82a]
The Software Engineering Laboratory, SEL-81-104, Software Engineering Laboratory Series, February 1982.

[SEL82b]
Guide to Data Collection, SEL-81-101, Software Engineering Laboratory Series, August 1982.

[Siegel56]
Siegel, Sidney, Nonparametric Statistics for the Behavioral Sciences, McGraw-Hill Book Company, Inc., New York, 1956.

[STL82]
Research and Development Mission Analysis System (RADMAS) System Description, STL-82-005, Systems Technology Laboratory Series, July 1982.

[Stucki77]
Stucki, Leon G., New Directions in Automated Tools for Improving Software Quality, pp. 80-111 in Current Trends in Programming Methodology, Vol II: Program Validation, ed. Raymond T. Yeh, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

NUMBER ____________

CHANGE REPORT FORM

PROJECT NAME ______________________________ CURRENT DATE __________

SECTION A - IDENTIFICATION

REASON: Why was the change made? ____________________

DESCRIPTION: What change was made? ____________________

EFFECT: What components (or documents) are changed? (Include version) ____________________

EFFORT: What additional components (or documents) were examined in determining what change was needed? ____________________

	(Month	Day	Year)
Need for change determined on			
Change started on			

What was the effort in person time required to understand and implement the change?

_____1 hour or less, _____1 hour to 1 day, _____1 day to 3 days, _____more than 3 days

SECTION B - TYPE OF CHANGE (How is this change best characterized?)

- ☐ Error correction
- ☐ Planned enhancement
- ☐ Implementation of requirements change
- ☐ Improvement of clarity, maintainability, or documentation
- ☐ Improvement of user services
- ☐ Insertion/deletion of debug code
- ☐ Optimization of time/space/accuracy
- ☐ Adaptation to environment change
- ☐ Other (Explain in E)

Was more than one component affected by the change? Yes_________ No_________

FOR ERROR CORRECTIONS ONLY

SECTION C - TYPE OF ERROR (How is this error best characterized?)

- ☐ Requirements incorrect or misinterpreted
- ☐ Functional specifications incorrect or misinterpreted
- ☐ Design error, involving several components
- ☐ Error in the design or implementation of a single component
- ☐ Misunderstanding of external environment, except language
- ☐ Error in use of programming language/compiler
- ☐ Clerical error
- ☐ Other (Explain in E)

FOR DESIGN OR IMPLEMENTATION ERRORS ONLY

→ If the error was in design or implementation:

The error was a mistaken assumption about the value or structure of data ____________

The error was a mistake in control logic or computation of an expression ____________

580-2 (6/78)

Figure 2.

FOR ERROR CORRECTIONS ONLY

SECTION D - VALIDATION AND REPAIR

What activities were used to validate the program, detect the error, and find its cause?

	Activities Used for Program Validation	Activities Successful in Detecting Error Symptoms	Activities Tried to Find Cause	Activities Successful in Finding Cause
Pre-acceptance test runs				
Acceptance testing				
Post-acceptance use				
Inspection of output				
Code reading by programmer				
Code reading by other person				
Talks with other programmers				
Special debug code				
System error messages				
Project specific error messages				
Reading documentation				
Trace				
Dump				
Cross-reference/attribute list				
Proof technique				
Other (Explain in E)				

What was the time used to isolate the cause?

____one hour or less, ____one hour to one day, ____more than one day, ____never found

If never found, was a workaround used?________Yes________No (Explain in E)

Was this error related to a previous change?

____Yes (Change Report #/Date__________) ____No ____Can't tell

When did the error enter the system?

____requirements ____functional specs ____design ____coding and test ____other ____can't tell

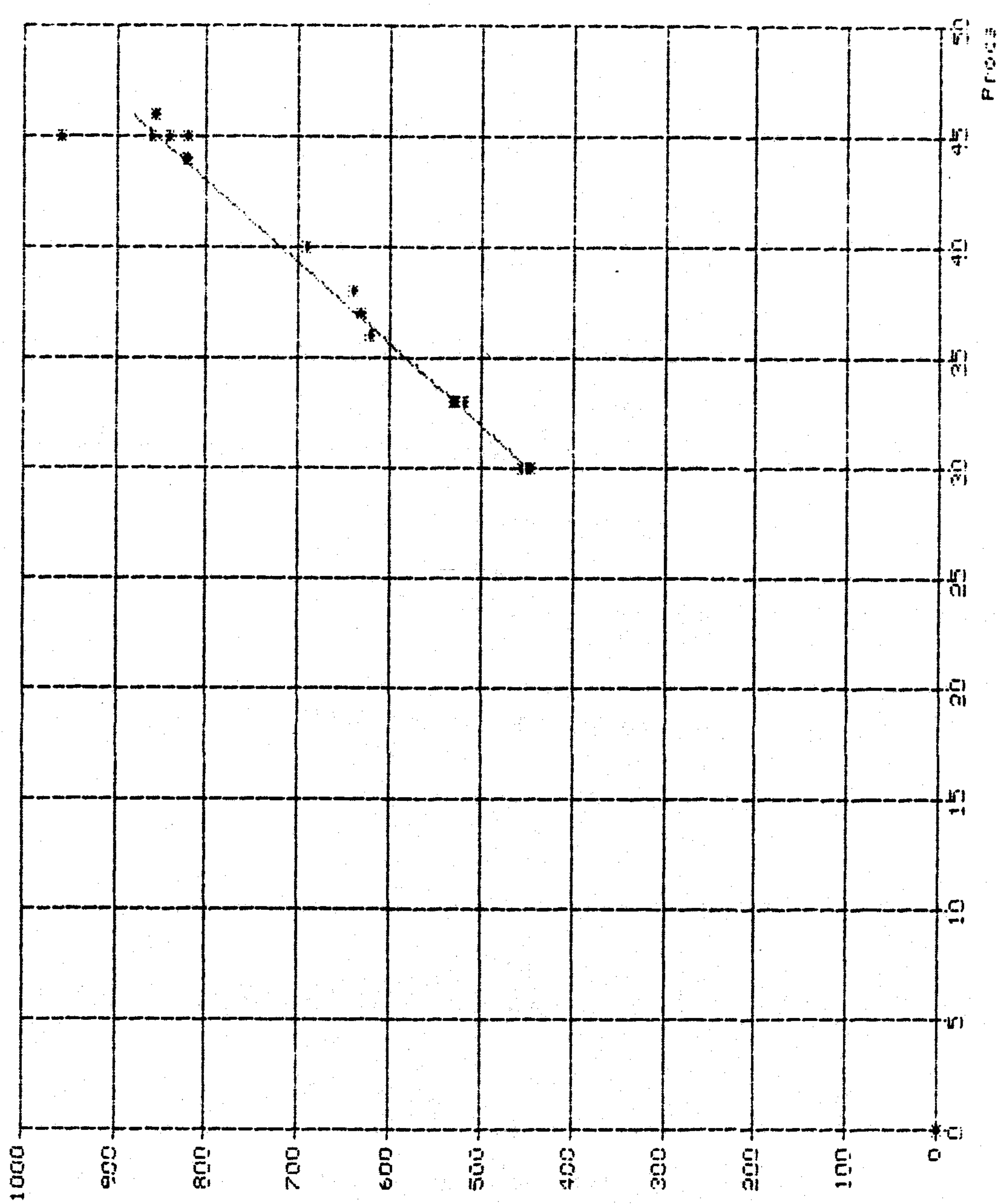
SECTION E - ADDITIONAL INFORMATION

Please give any information that may be helpful in categorizing the error or change, and understanding its cause and its ramifications.

Name: ______________________ Authorized: ______________________ Date: __________

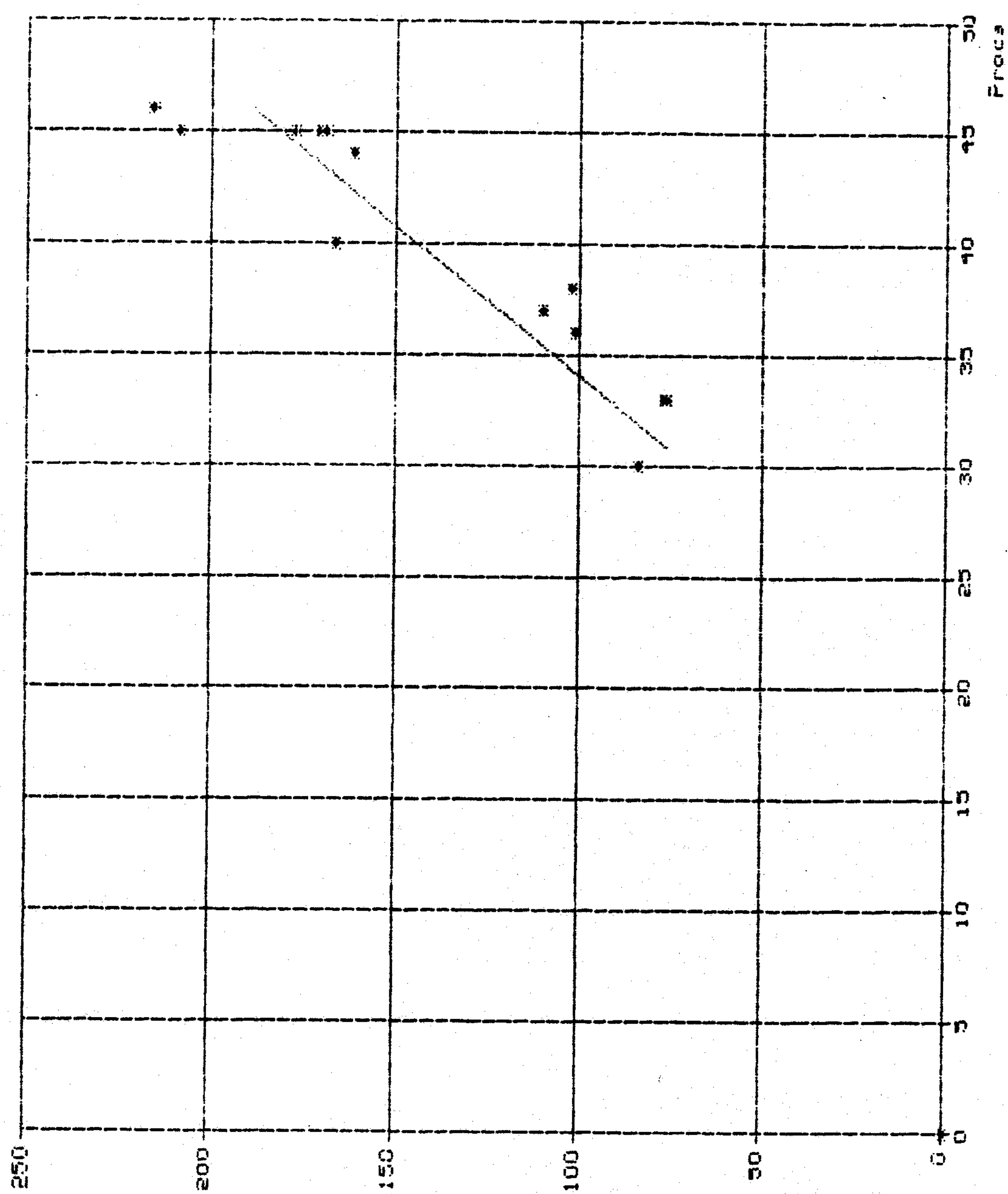
580-2 (5/78)

Plot 1. Assignment Statement Coverage vs Procedure Coverage for 17 Acceptance Tests.



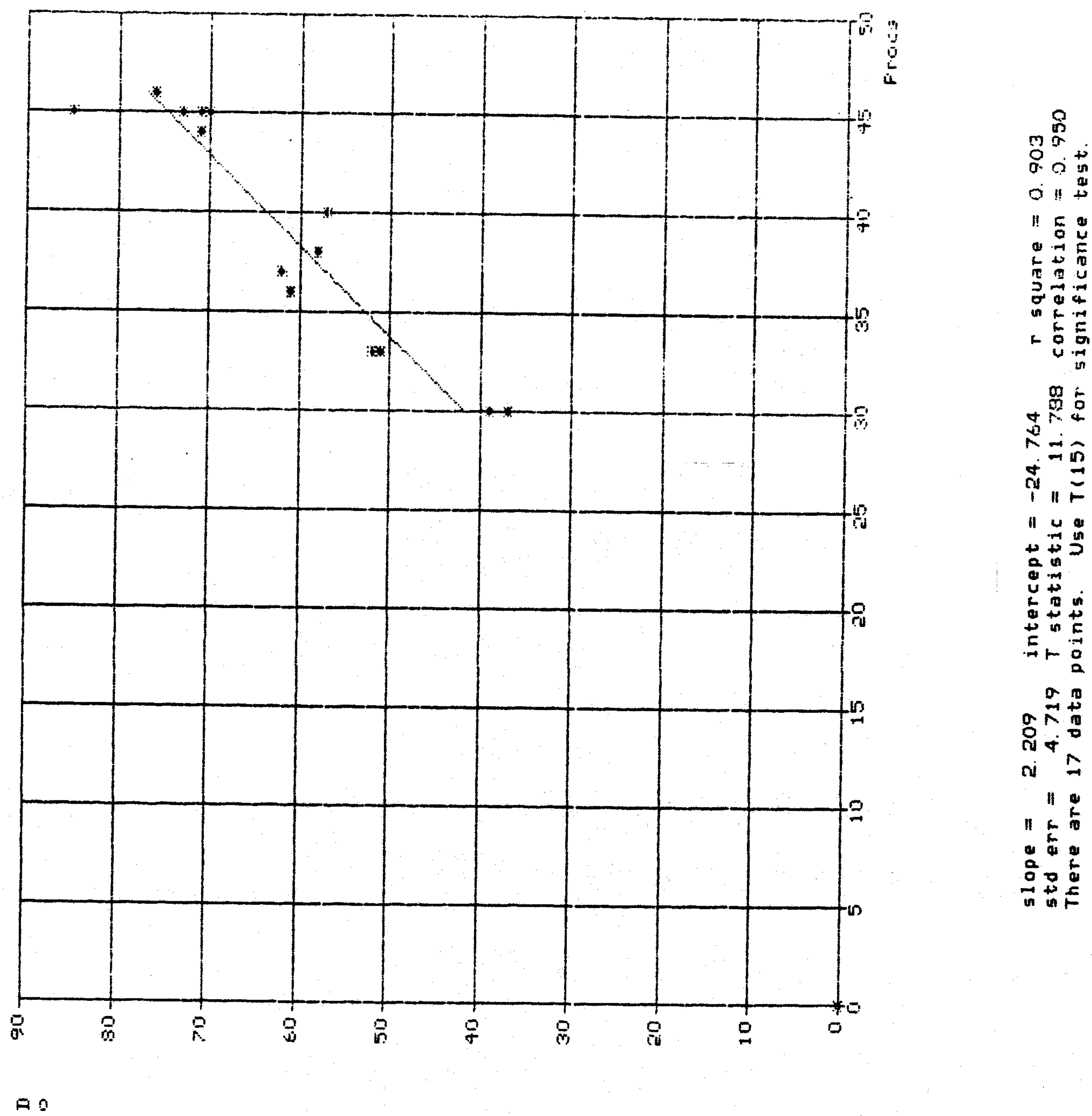
slope = 27.059 intercept = -364.122 r square = 0.970
std err = 31.191 T statistic = 21.842 correlation = 0.985
There are 17 data points. Use T(15) for significance test.

Plot 2. CALL Statement Coverage vs Procedure Coverage for 17 Acceptance Tests.

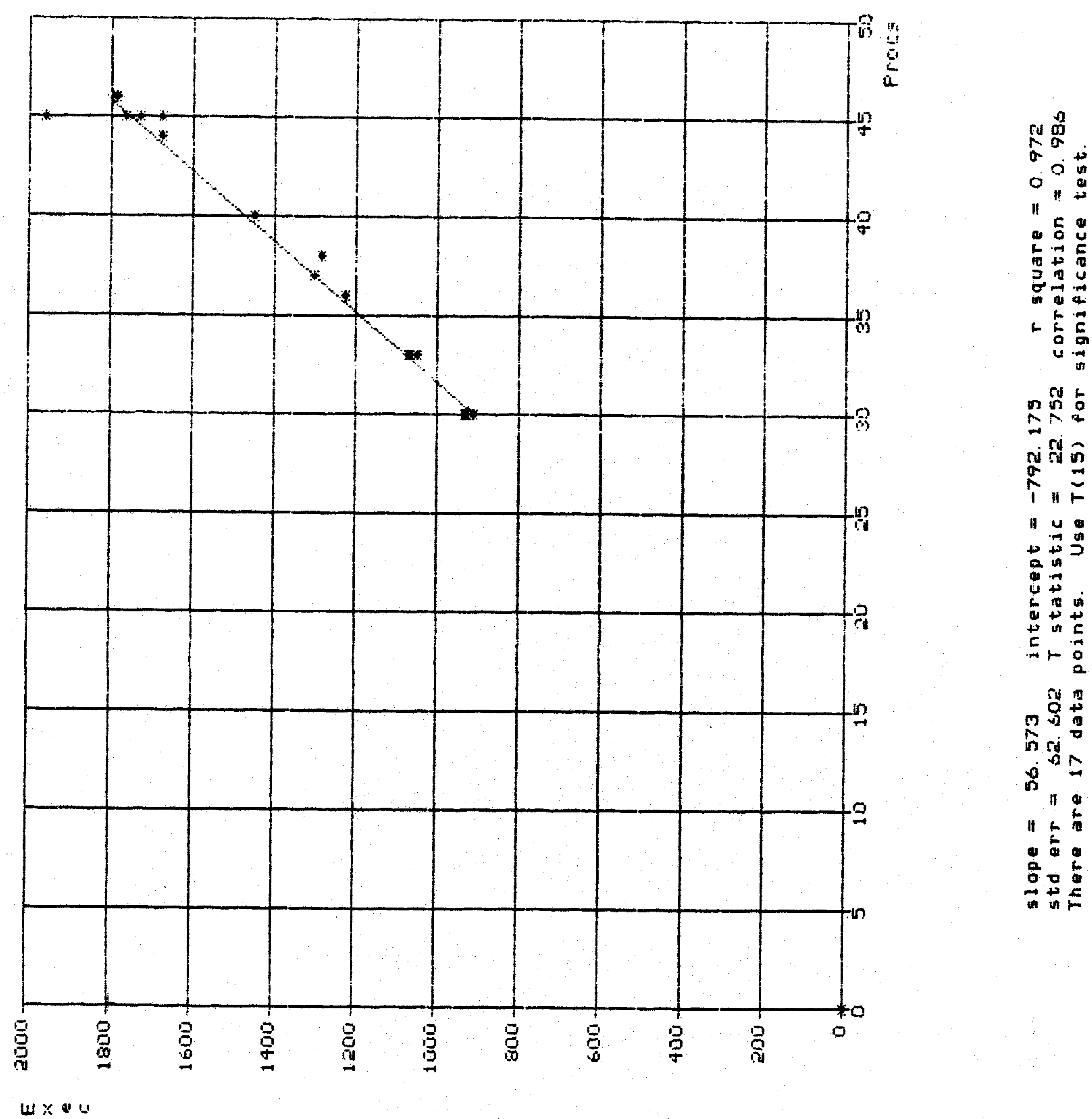


slope = 7.447 intercept = -154.078 r square = 0.875
std err = 18.338 T statistic = 10.225 correlation = 0.935
There are 17 data points. Use T(15) for significance test.

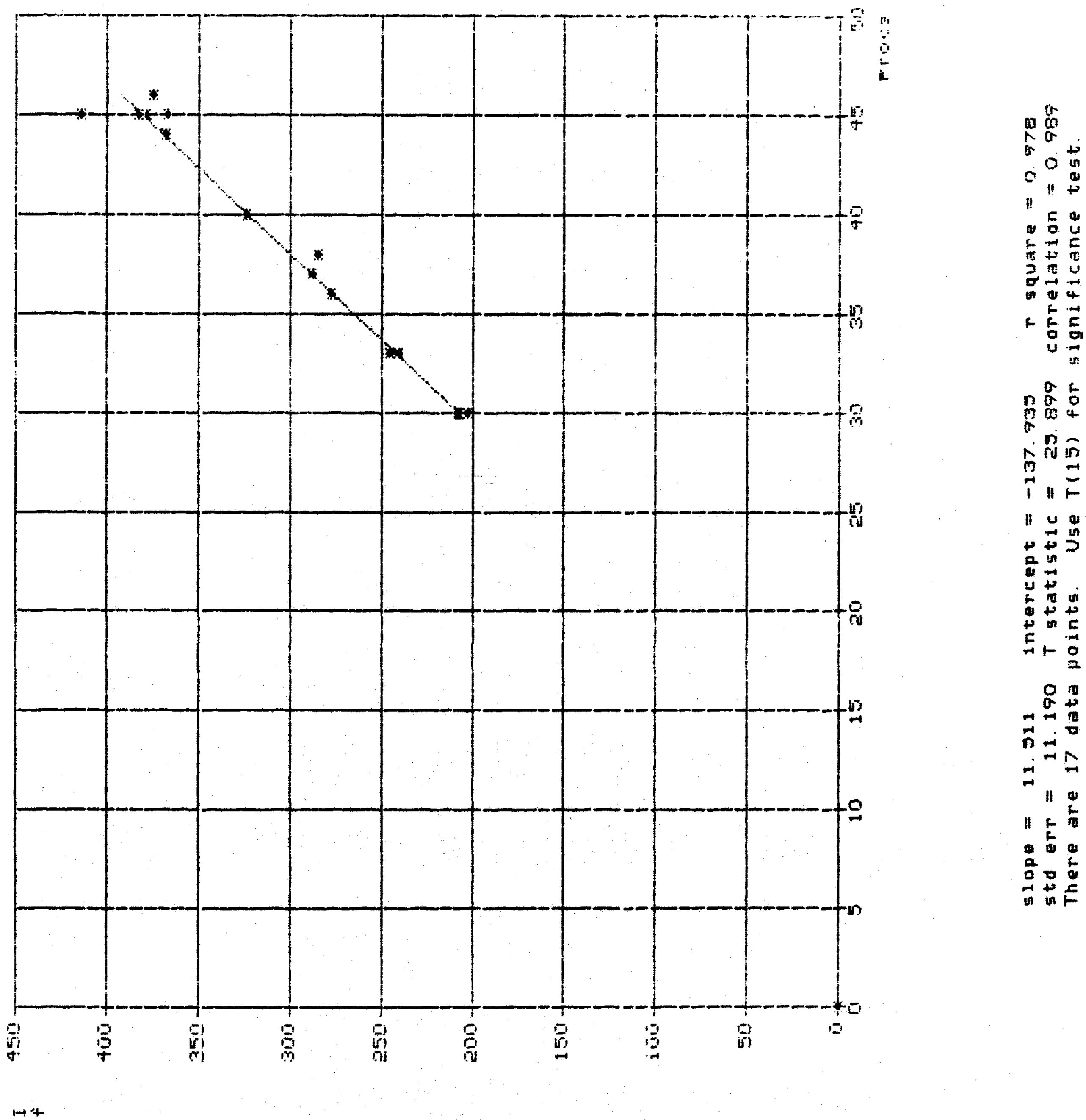
Plot 3. Do Statement Coverage vs Procedure Coverage for 17 Acceptance Tests.



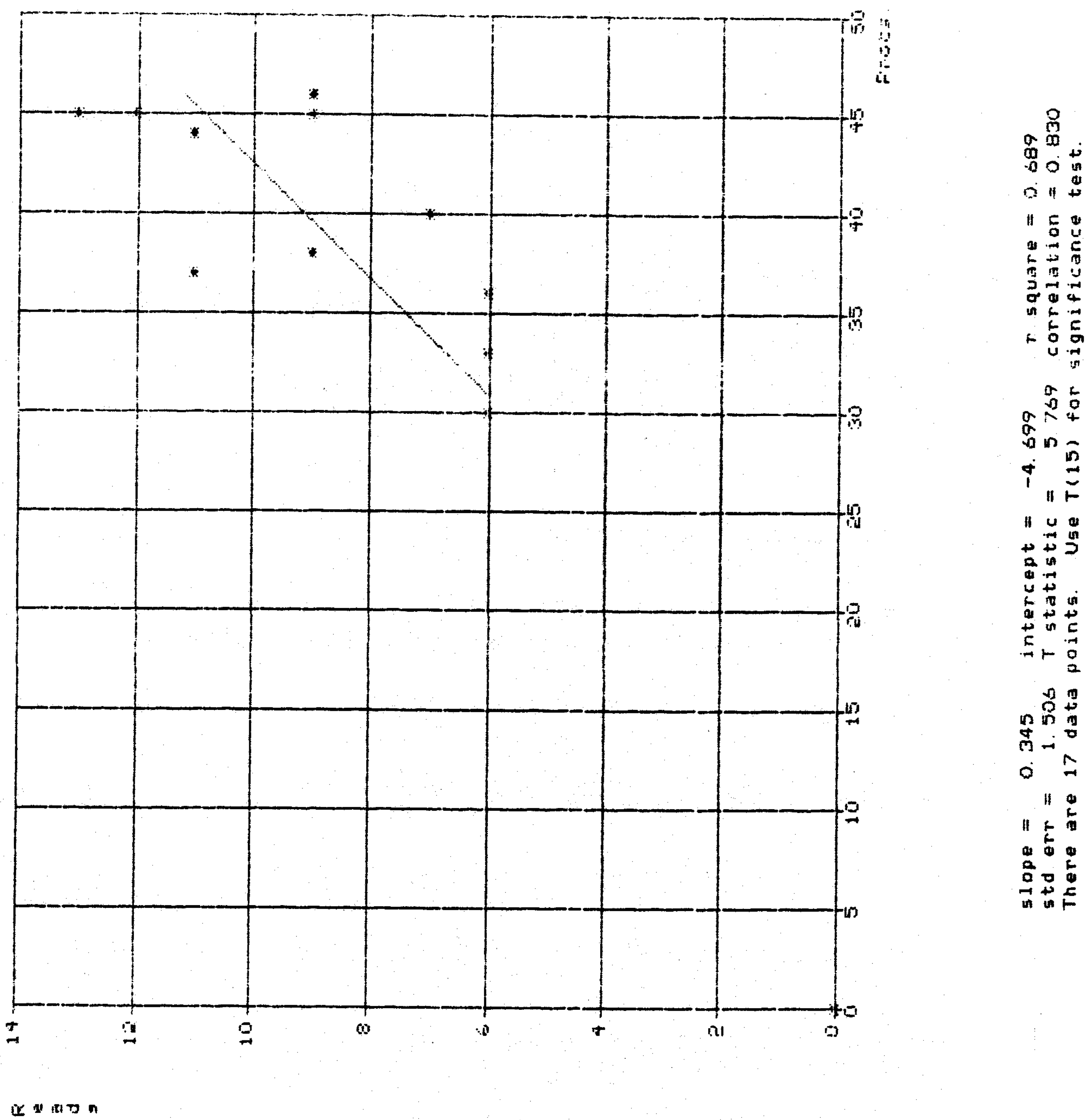
Plot 4. Executable Statement Coverage vs Procedure Coverage for 17 Acceptance Tests.



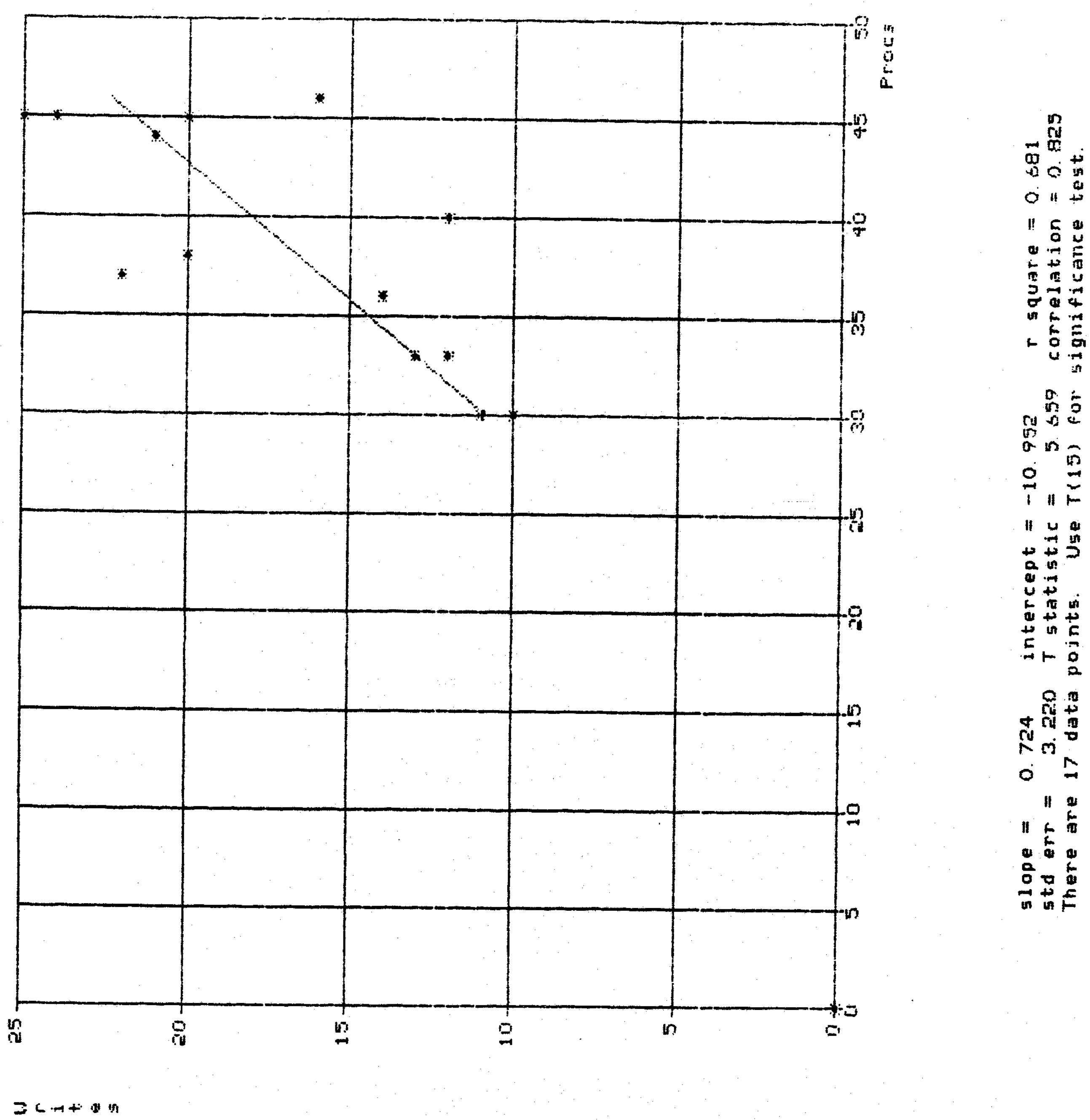
Plot 5. IF Statement Coverage vs Procedure Coverage for 17 Acceptance Tests.



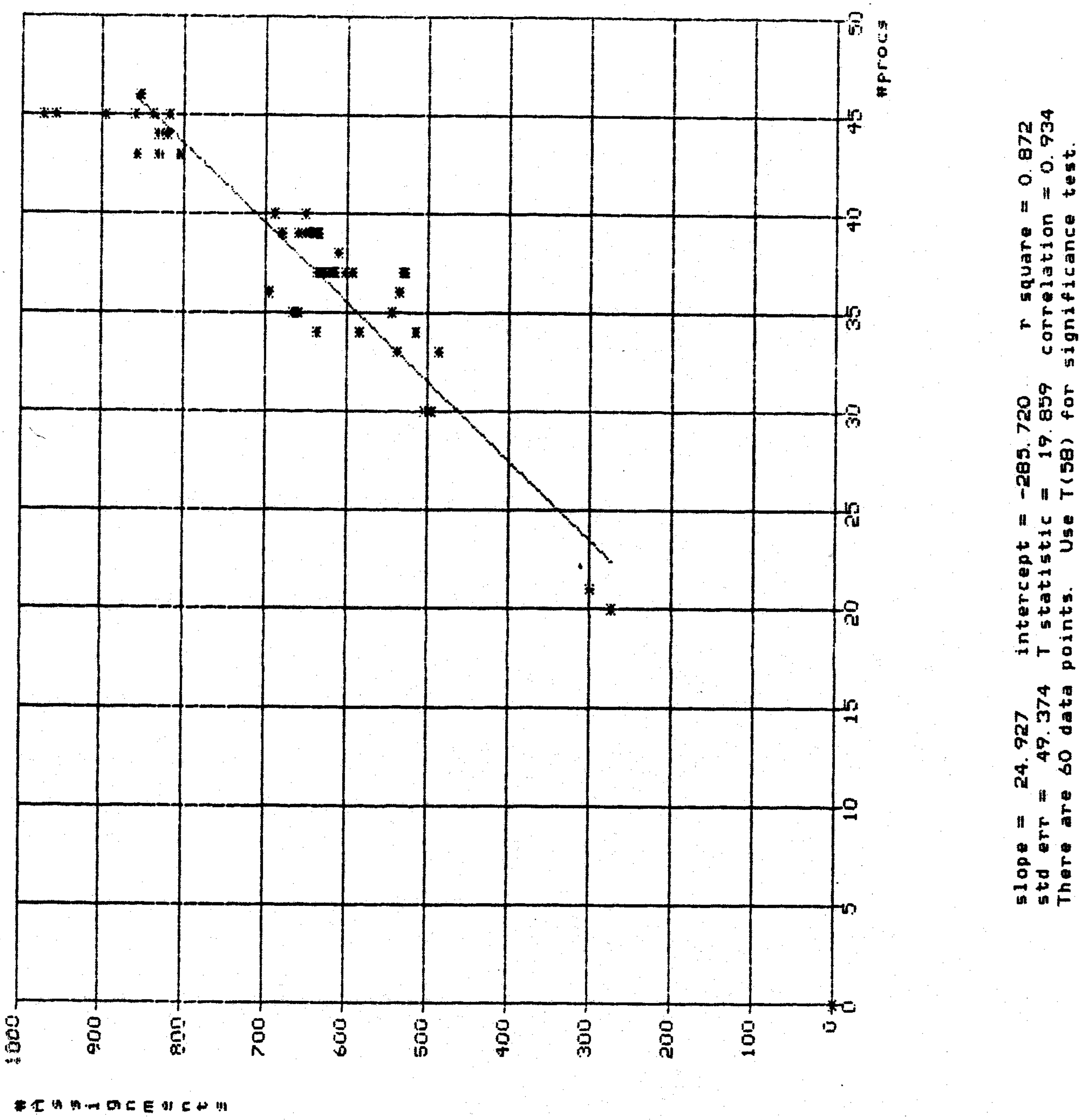
Plot 6. READ Statement Coverage vs Procedure Coverage for 17 Acceptance Tests.



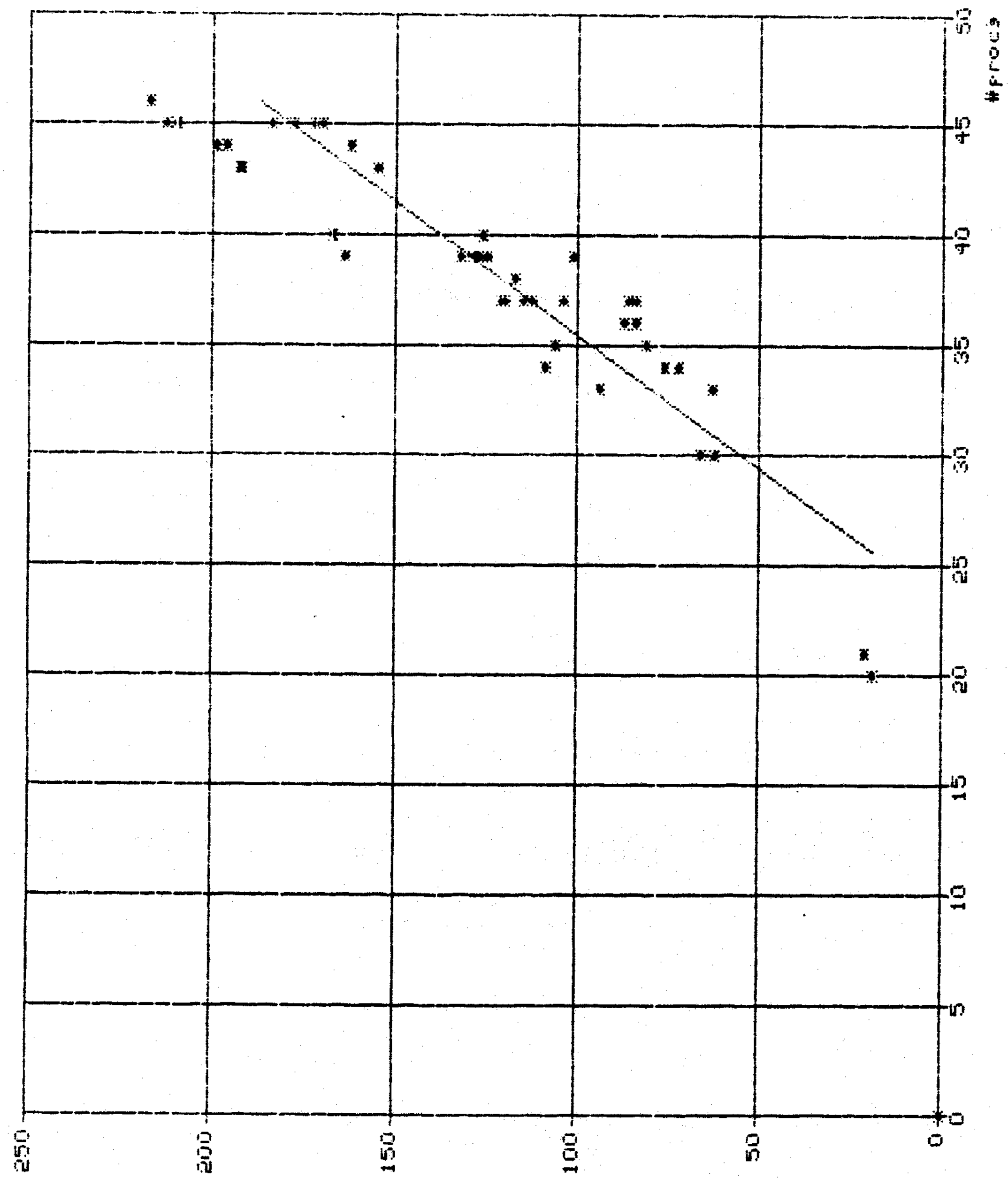
Plot 7. WRITE Statement Coverage vs Procedure Coverage for 17 Acceptance Tests.



Plot 8. Assignment Statement Coverage vs Procedure Coverage for 60 Operational Usage Cases.

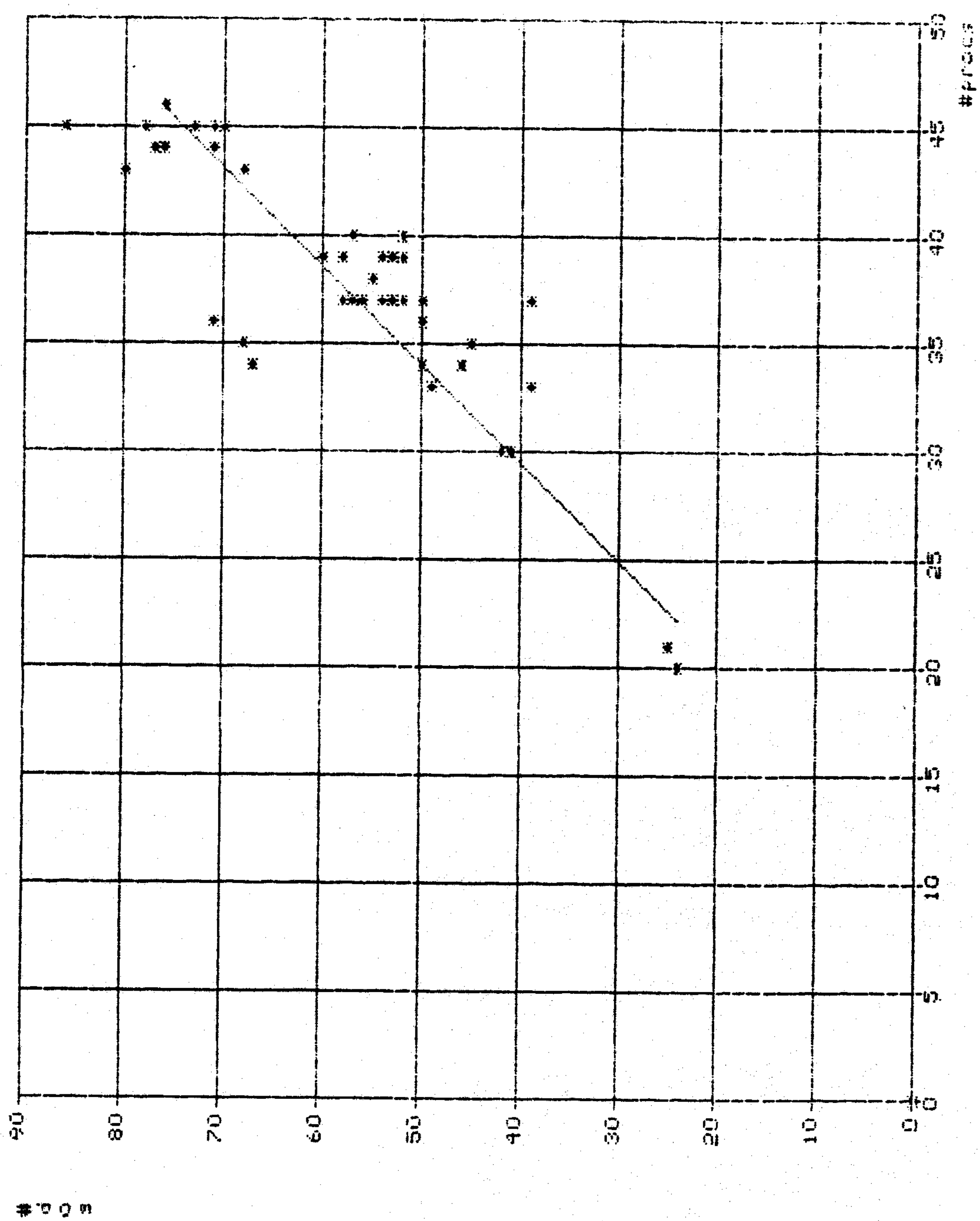


Plot 9. CALL Statement Coverage vs Procedure Coverage for 60 Operational Usage Cases.



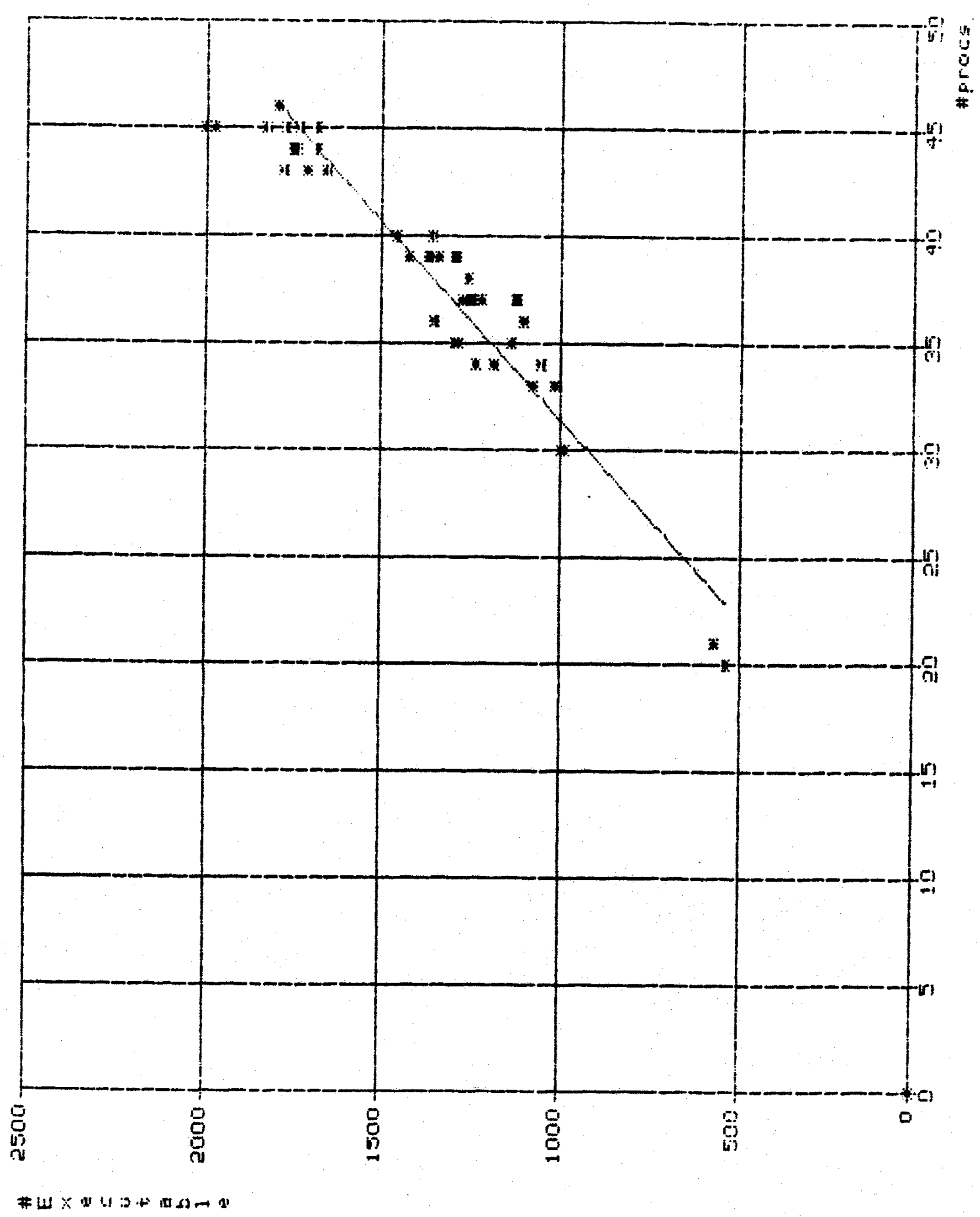
slope = 8.251 intercept = -192.712 r square = 0.799
std err = 21.364 T statistic = 15.192 correlation = 0.894
There are 60 data points. Use T(58) for significance test.

Plot 10. Do Statement Coverage vs Procedure Coverage for 60 Operational Usage Cases.



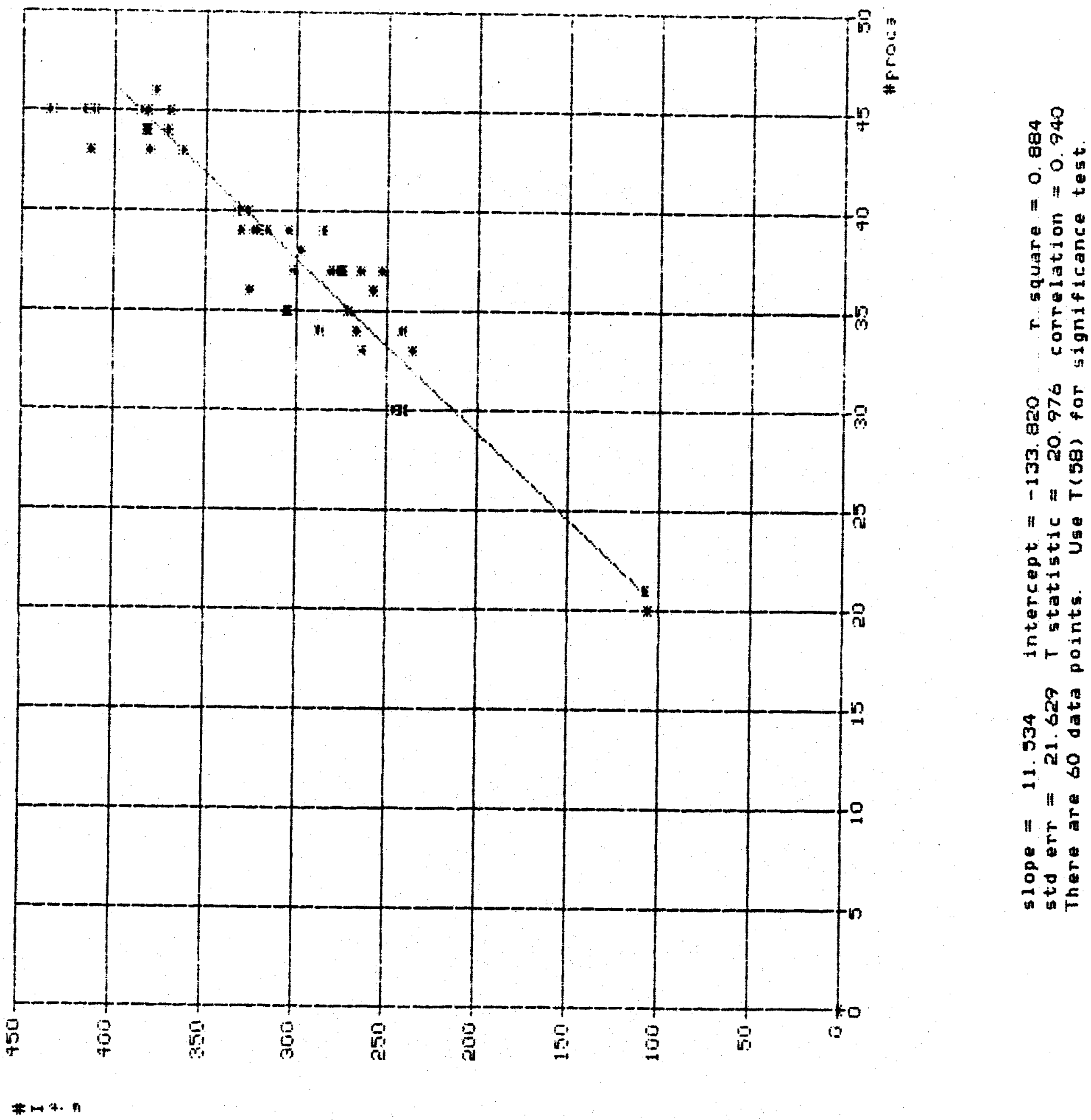
slope = 2.197 intercept = -24.887 r square = 0.728
std err = 6.939 T statistic = 12.457 correlation = 0.853
There are 60 data points. Use T(58) for significance test.

Plot 11. Executable Statement Coverage vs Procedure Coverage for 60 Operational Usage Cases.

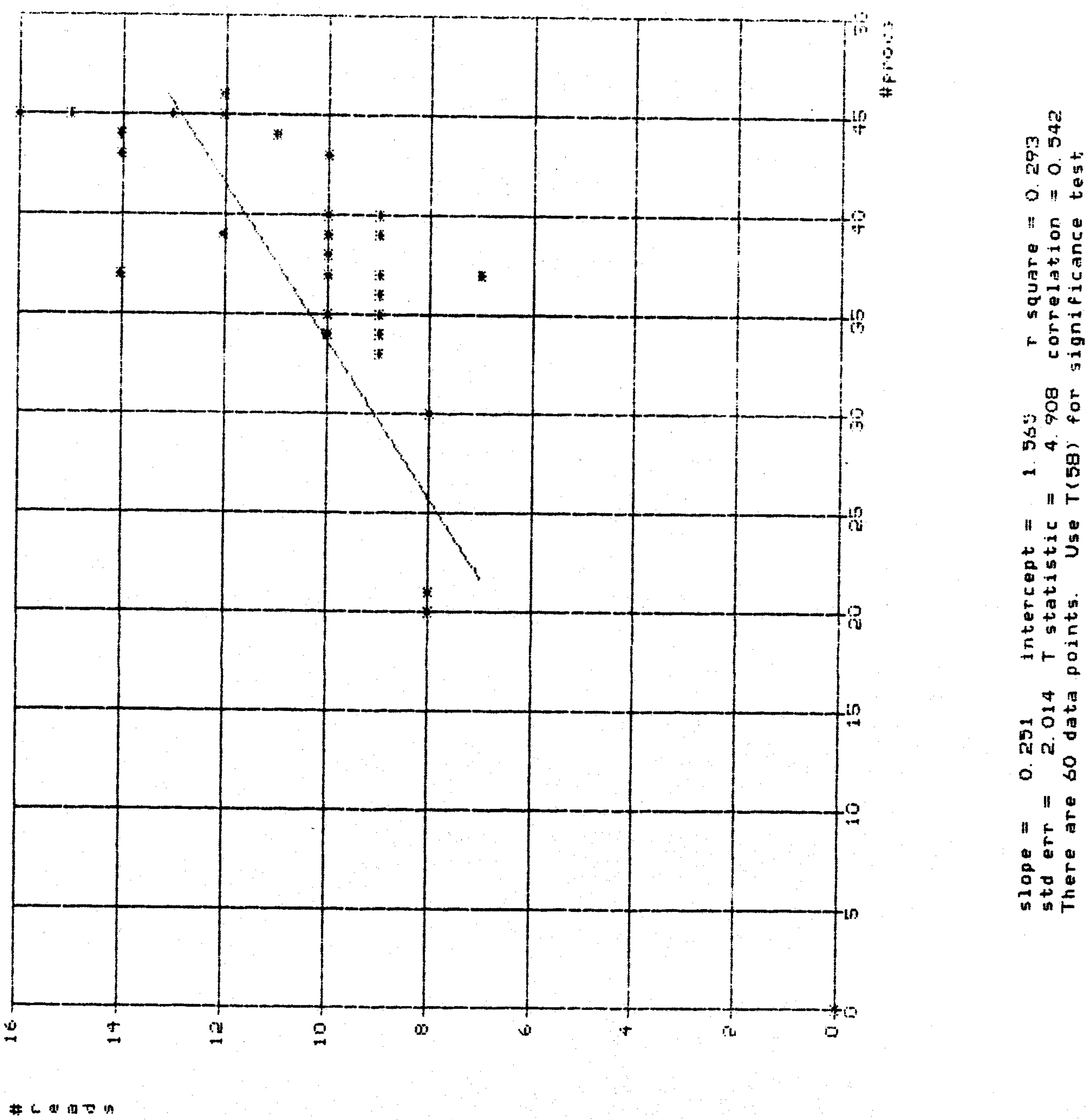


slope = 54.558 intercept = -722.445 r square = 0.905
std err = 91.505 T statistic = 23.453 correlation = 0.951
There are 60 data points. Use T(58) for significance test.

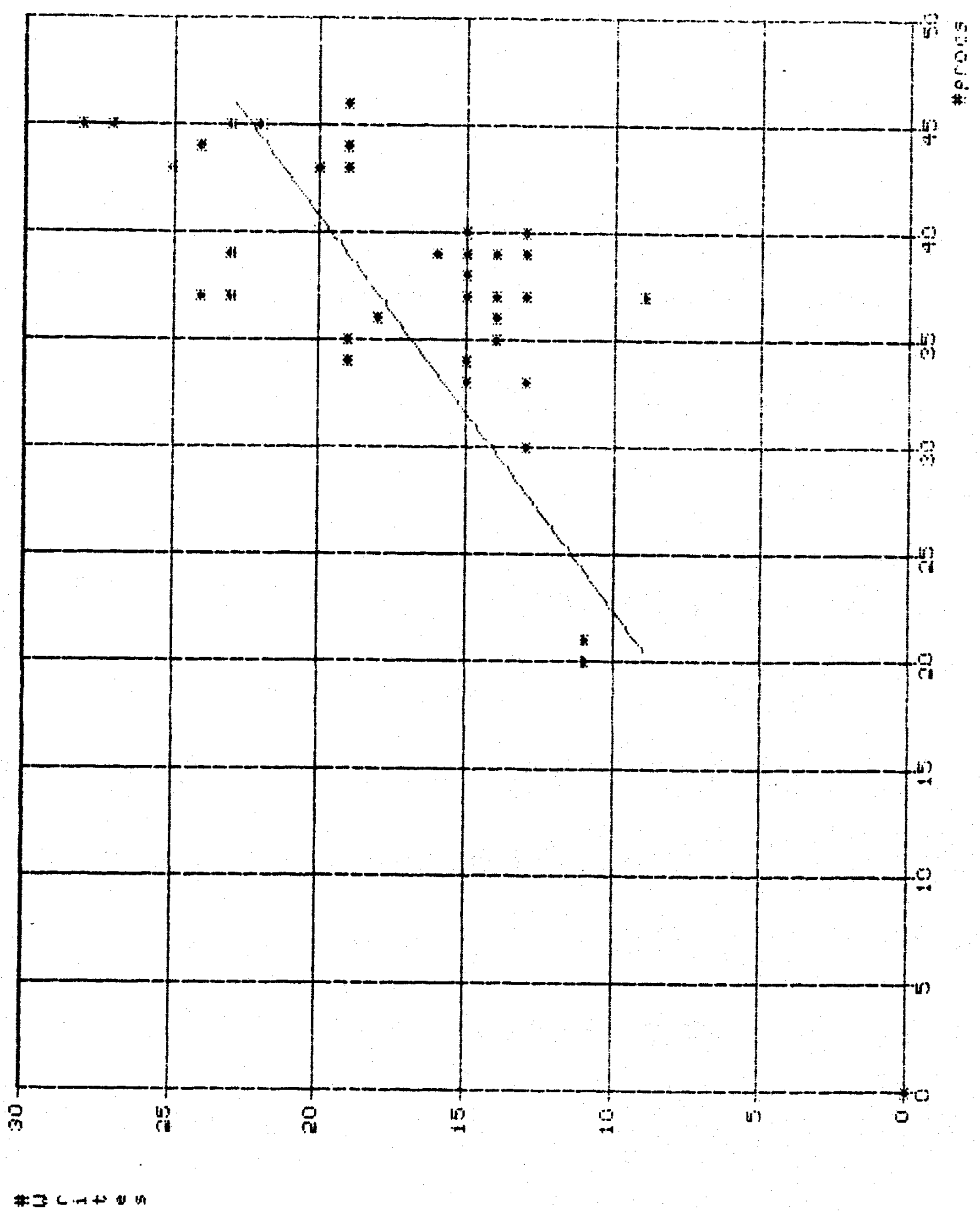
Plot 12. IF Statement Coverage vs Procedure Coverage for 60 Operational Usage Cases.



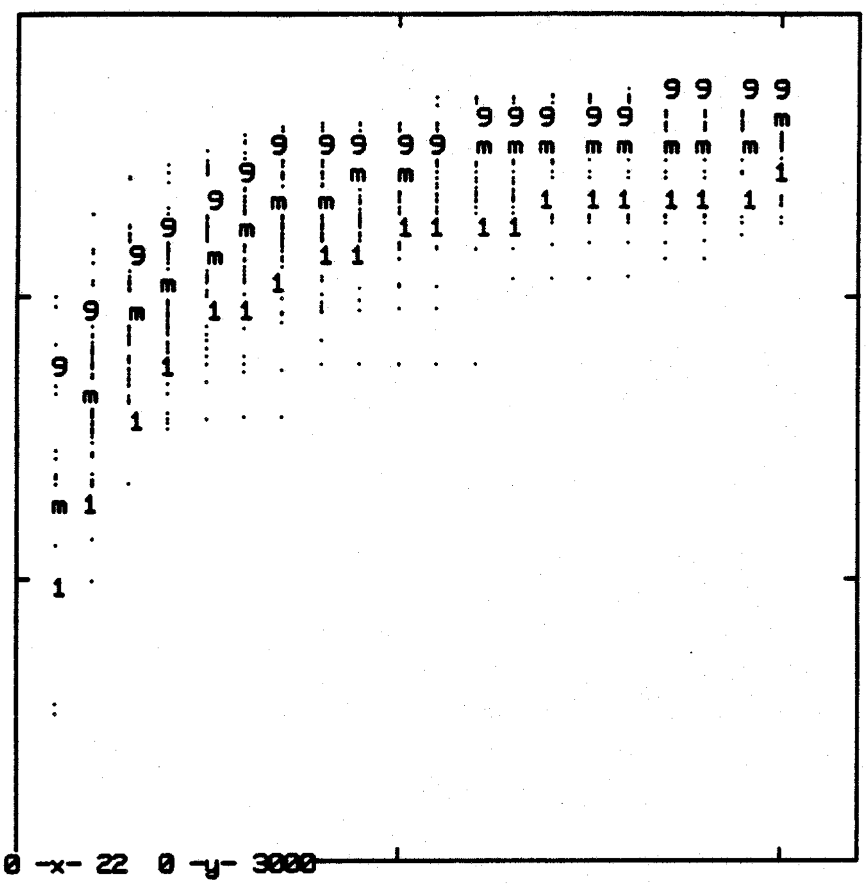
Plot 13. READ Statement Coverage vs Procedure Coverage for 60 Operational Usage Cases.



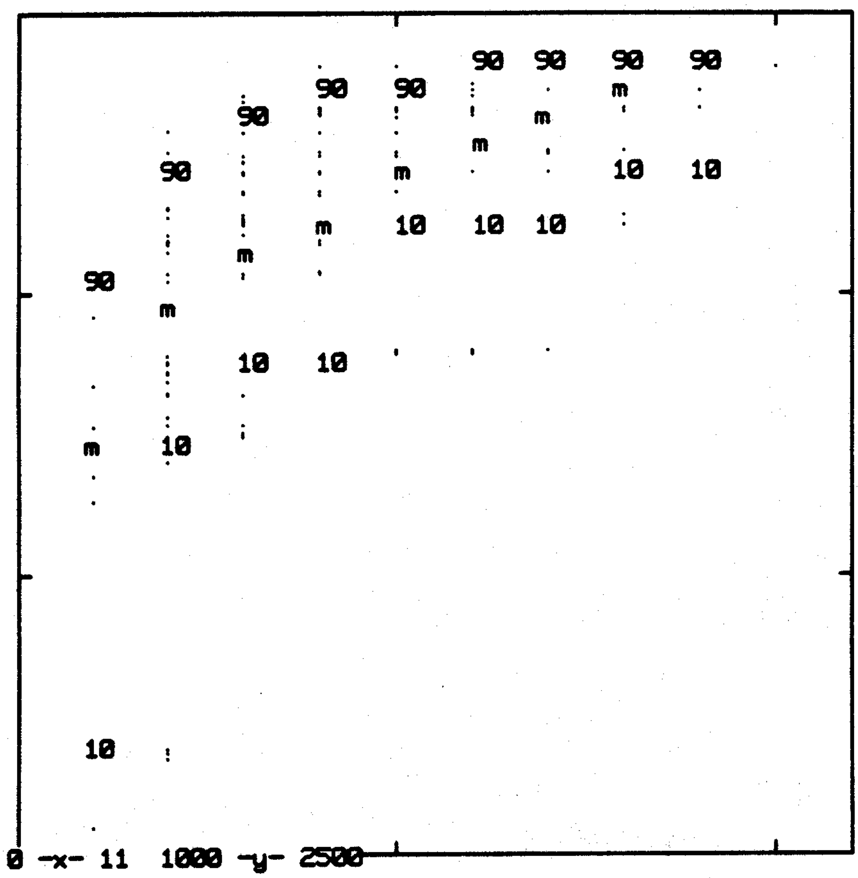
Plot 14. WRITE Statement Coverage vs Procedure Coverage for 60 Operational Usage Cases.



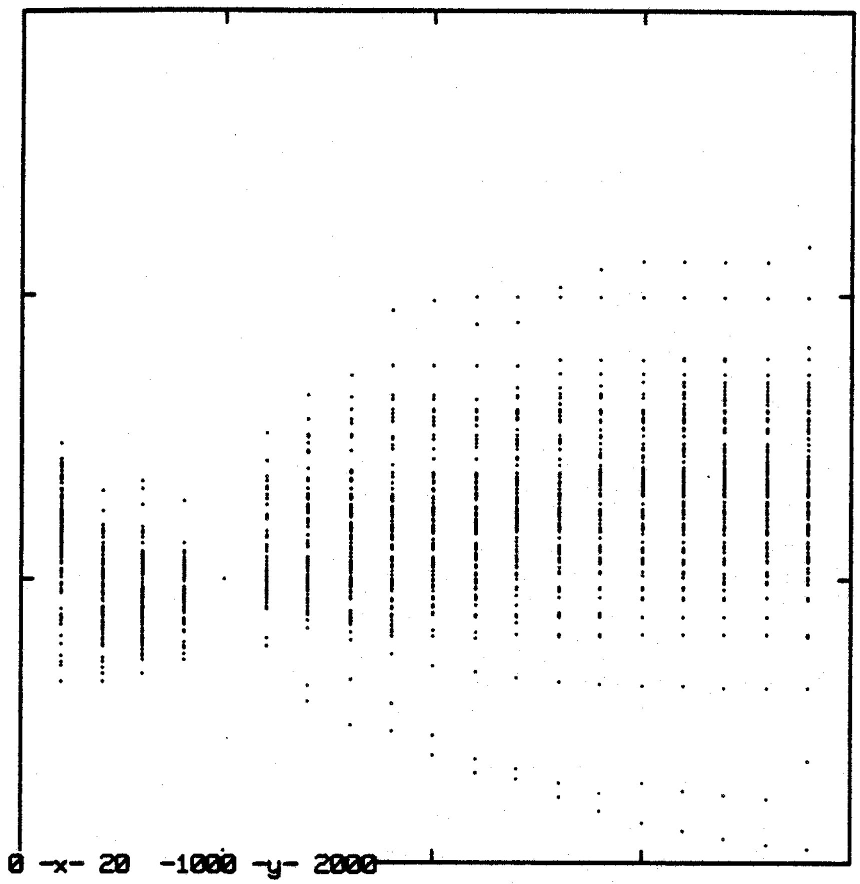
slope = 0.543 intercept = -2.116 r square = 0.313
std err = 4.153 T statistic = 5.139 correlation = 0.559
There are 60 data points. Use T(58) for significance test



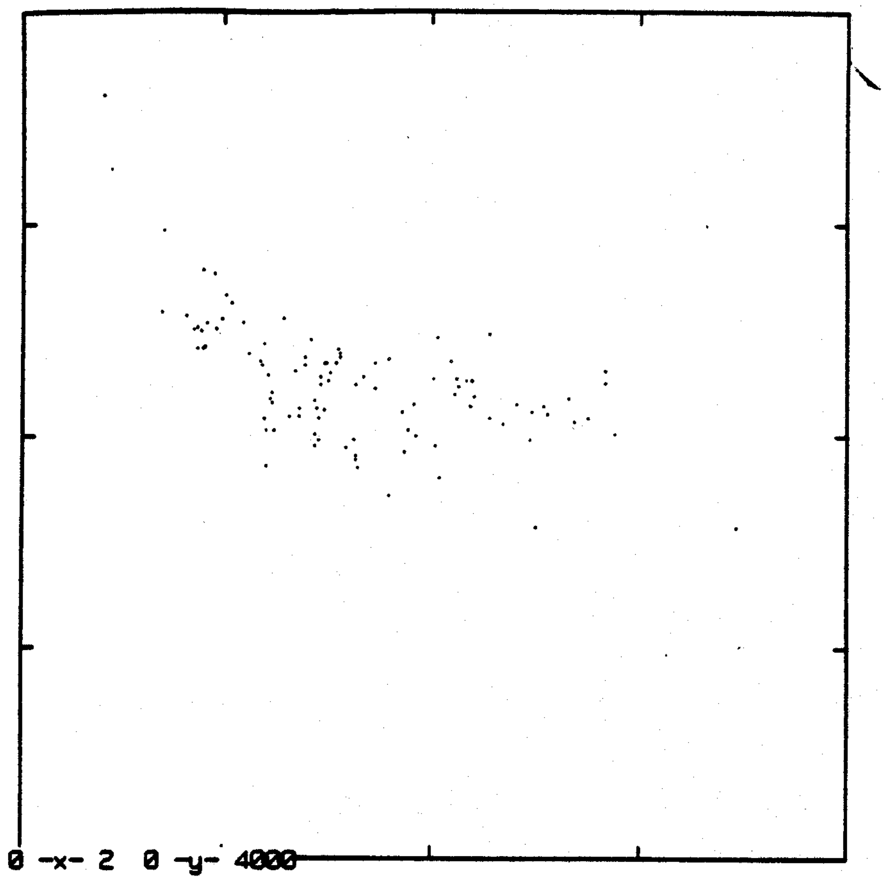
Plot 15. Structural Coverage of 100 Permutations of 60 Operational Usage Cases. (median, 10th, and 90th percentiles superimposed)



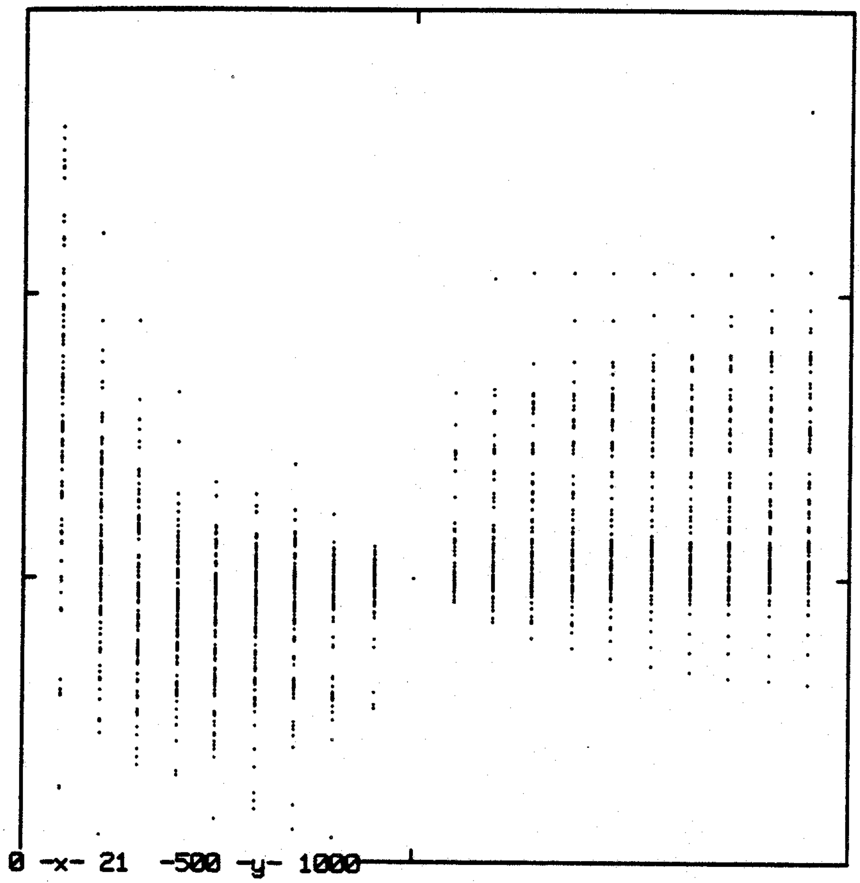
Plot 16. Structural Coverage of 100 Permutations of 10 Acceptance Tests. (median, 10th, and 90th percentiles superimposed)



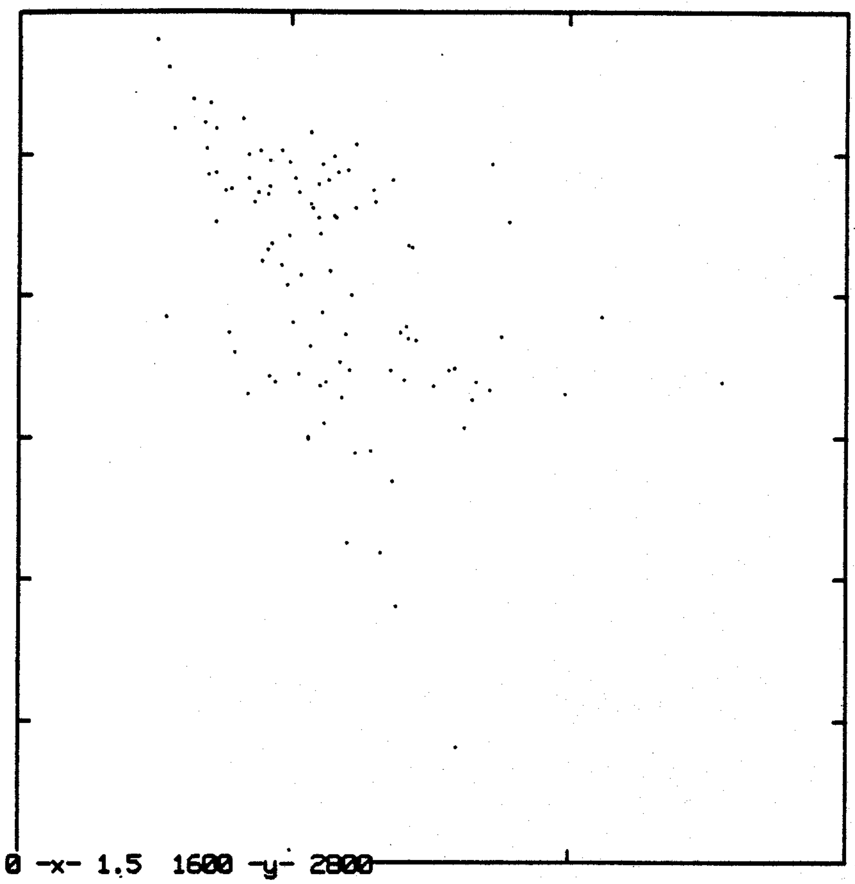
Plot 17. NHPP Model Fitted to the First 5 Values of the 100 Operational Growth Sequences. (residuals)



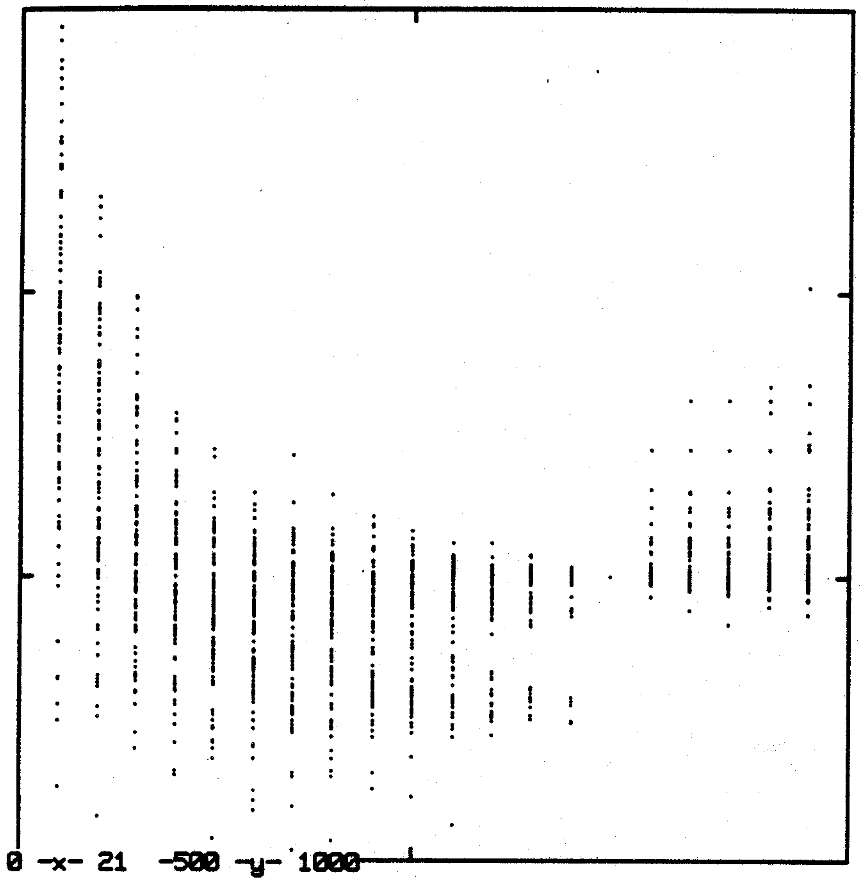
Plot 18. NHPP Model Fitted to the First 5 Values of the 100 Operational Growth Sequences. (plot of a vs b)



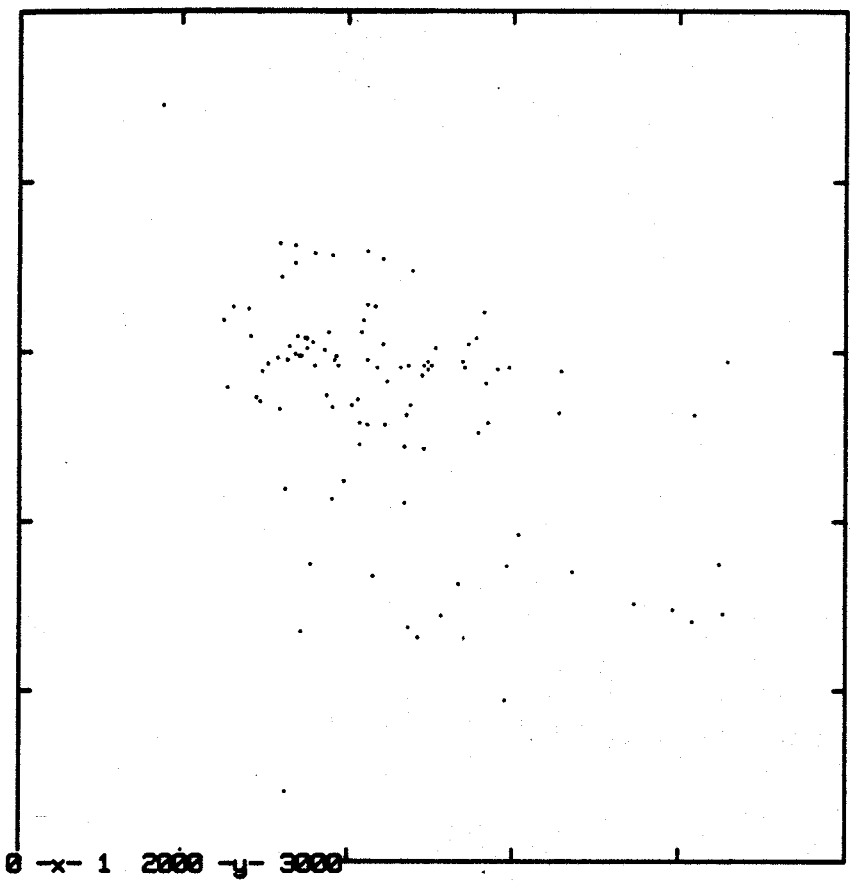
Plot 19. NHPP Model Fitted to the First 10 Values of the 100 Operational Growth Sequences. (residuals)



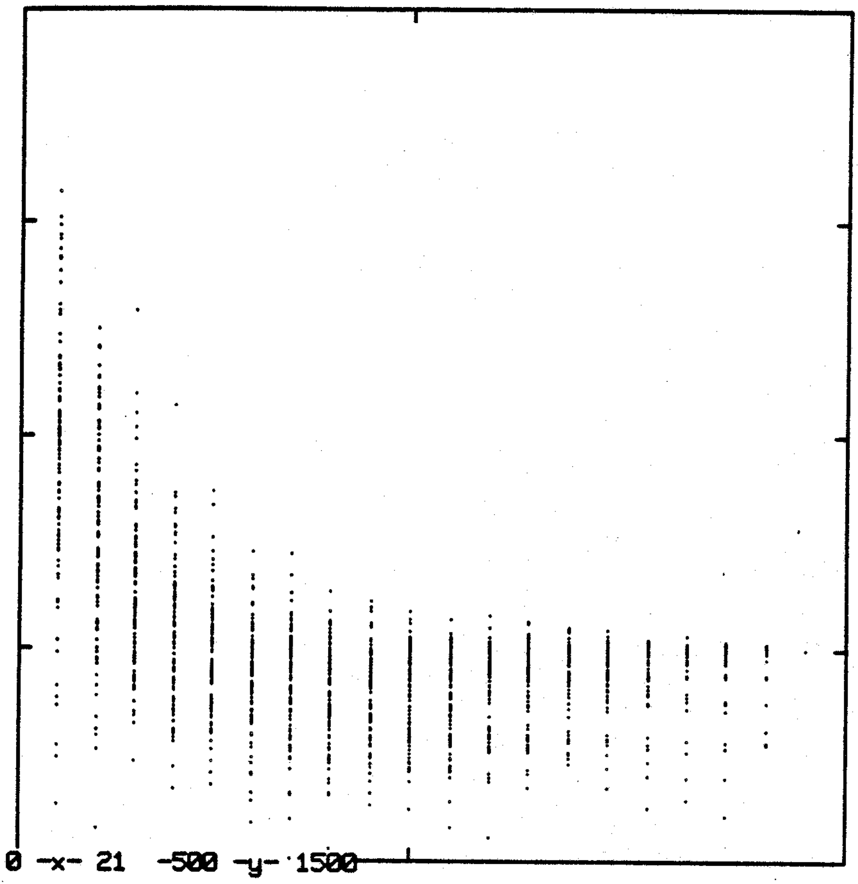
Plot 20. NHPP Model Fitted to the First 10 Values of the 100 Operational Growth Sequences. (plot of a vs b)



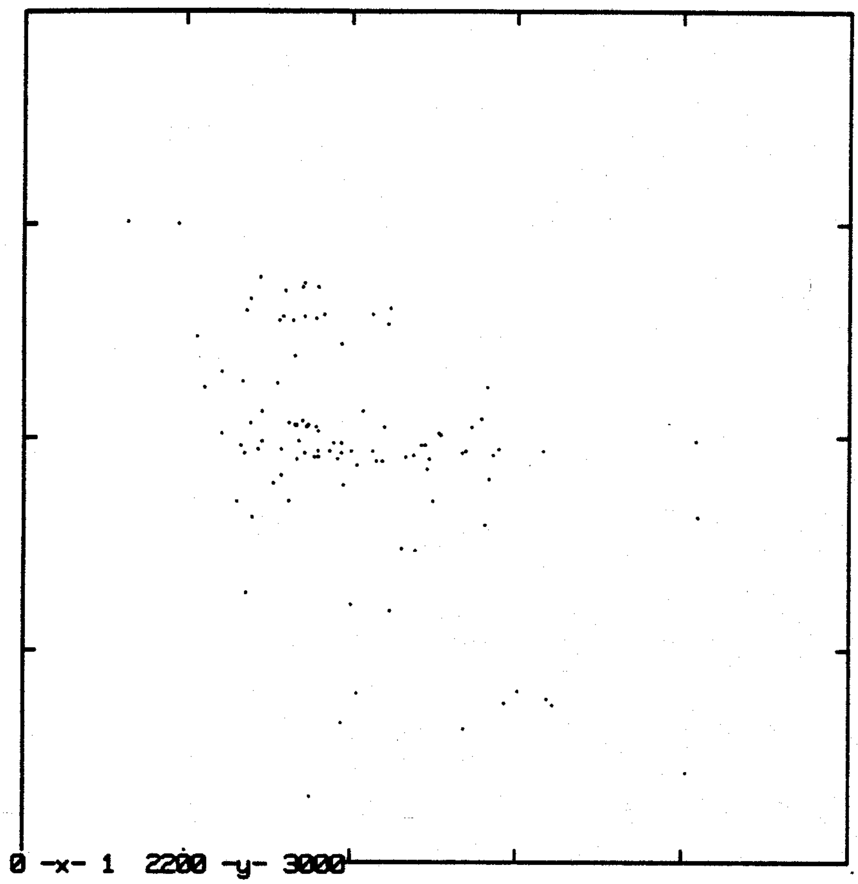
Plot 21. NHPP Model Fitted to the First 15 Values of the 100 Operational Growth Sequences. (residuals)



Plot 22. NHPP Model Fitted to the First 15 Values of the 100 Operational Growth Sequences. (plot of a vs b)

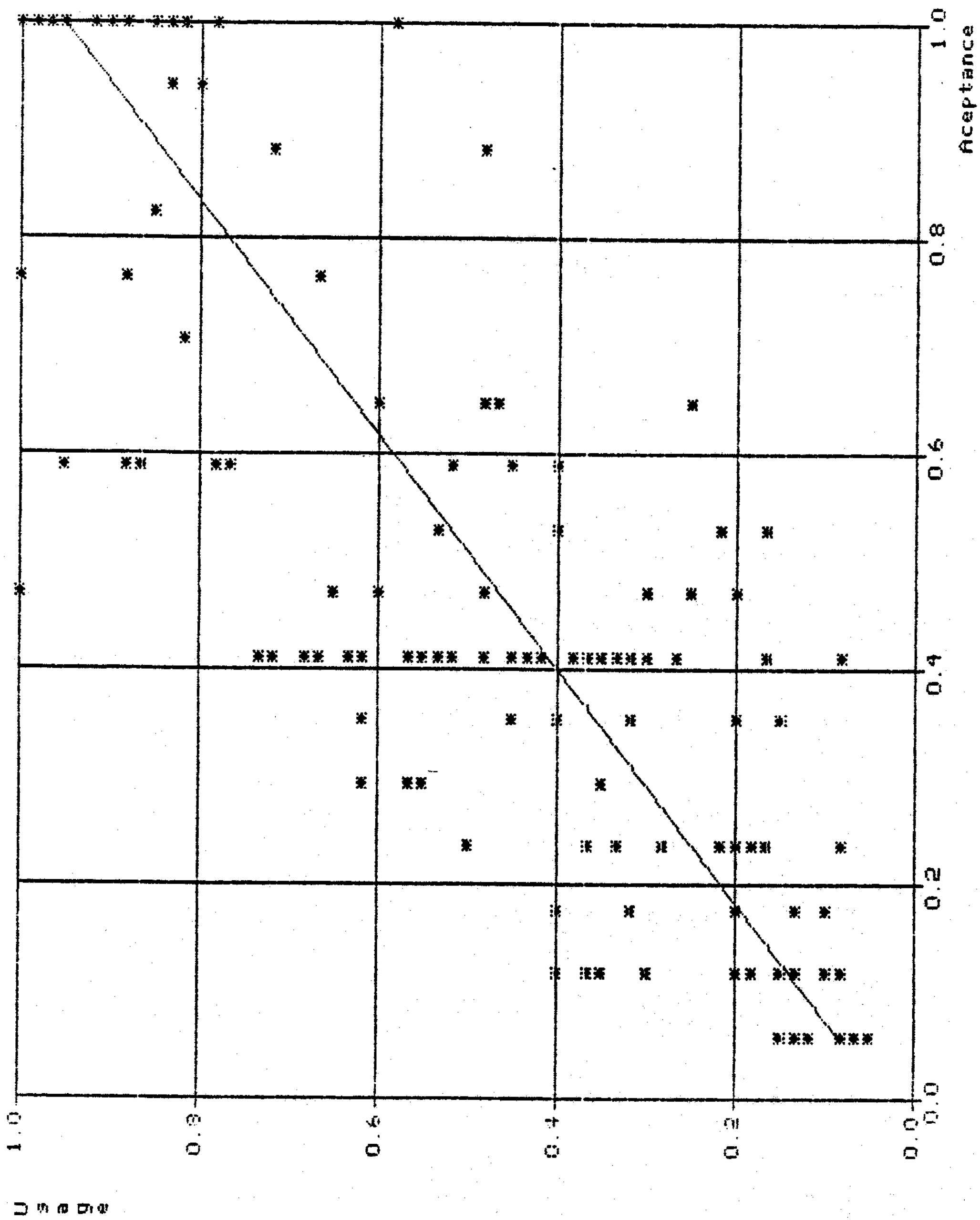


Plot 23. NHPP Model Fitted to the First 20 Values of the 100 Operational Growth Sequences. (residuals)



Plot 24. NHPP Model Fitted to the First 20 Values of the 100 Operational Growth Sequences. (plot of a vs b)

Plot 25. Comparison of Execution Coverage of Acceptance Test and Operational Usage.



slope = 0.921 intercept = 0.032 r square = 0.863
std err = 0.129 T statistic = 83.652 correlation = 0.929
There are 1115 data points. Use T(1113) for significance test.

Statement Coverage of the MAL Preprocessor by 17 Benchmark Test Cases.								
Case	Procs	Exec	Assign	Calls	Do	If	Reads	Writes
1	33	1069	530	76	52	246	6	13
2	30	913	446	84	37	203	6	10
3	33	1067	529	76	52	246	6	13
4	30	932	456	84	39	209	6	11
5	33	1049	519	77	51	241	6	12
6	37	1304	632	110	62	288	11	22
7	30	928	455	84	39	208	6	10
8	36	1228	622	101	61	278	6	14
9	30	928	455	84	39	208	6	10
10	44	1677	821	161	71	368	11	21
11	46	1786	855	216	76	375	9	16
12	38	1285	640	102	58	285	9	20
13	40	1448	691	166	57	324	7	12
14	45	1675	819	169	70	367	9	20
15	45	1959	957	209	85	414	13	25
16	45	1764	861	177	73	383	12	24
17	45	1728	840	171	71	379	12	24
Union	51	2408	1187	286	108	490	14	30
Intersect	29	778	389	42	35	186	6	10
Maximum	68	4300	1870	418	157	753	34	206

Table 1.

Statement Coverage
of the MAL Preprocessor
by 17 Benchmark Test Cases.
(Percentage of Maximum)

Case	Procs	Exec	Assign	Calls	Do	If	Reads	Writes
1	48.5	24.9	28.3	18.2	33.1	32.7	17.6	6.3
2	44.1	21.2	23.9	20.1	23.6	27.0	17.6	4.9
3	48.5	24.8	28.3	18.2	33.1	32.7	17.6	6.3
4	44.1	21.7	24.4	20.1	24.8	27.8	17.6	5.3
5	48.5	24.4	27.8	18.4	32.5	32.0	17.6	5.8
6	54.4	30.3	33.8	26.3	39.5	38.2	32.4	10.7
7	44.1	21.6	24.3	20.1	24.8	27.6	17.6	4.9
8	52.9	28.6	33.3	24.2	38.9	36.9	17.6	6.8
9	44.1	21.6	24.3	20.1	24.8	27.6	17.6	4.9
10	64.7	39.0	43.9	38.5	45.2	48.9	32.4	10.2
11	67.6	41.5	45.7	51.7	48.4	49.8	26.5	7.8
12	55.9	29.9	34.2	24.4	36.9	37.8	26.5	9.7
13	58.8	33.7	37.0	39.7	36.3	43.0	20.6	5.8
14	66.2	39.0	43.8	40.4	44.6	48.7	26.5	9.7
15	66.2	45.6	51.2	50.0	54.1	55.0	38.2	12.1
16	66.2	41.0	46.0	42.3	46.5	50.9	35.3	11.7
17	66.2	40.2	44.9	40.9	45.2	50.3	35.3	11.7
Union	75.0	56.0	63.5	68.4	68.8	65.1	41.2	14.6
Intersect	42.6	18.1	20.8	10.0	22.3	24.7	17.6	4.9

Table 2.

Statement Coverage of the MAL Preprocessor by 60 Operational Useage Cases.								
Case	Procs	Exec	Assign	Calls	Do	If	Reads	Writes
1	39	1368	660	125	52	324	10	14
2	43	1712	832	193	80	381	10	19
3	45	1830	896	184	78	411	13	22
4	37	1262	625	86	57	278	14	24
5	37	1252	618	120	53	276	10	15
6	36	1098	536	84	50	258	9	14
7	33	1012	486	94	39	236	9	13
8	39	1359	652	129	52	331	9	13
9	37	1246	619	84	56	275	14	23
10	37	1252	618	120	53	276	10	15
11	44	1743	831	196	76	382	11	19
12	37	1249	616	120	53	275	10	15
13	35	1295	666	81	68	306	10	19
14	35	1286	660	81	68	305	10	19
15	35	1136	545	106	45	272	9	14
16	46	1794	853	217	76	378	12	19
17	37	1272	637	86	57	278	14	24
18	37	1252	618	120	53	276	10	15
19	37	1270	635	86	57	278	14	24
20	37	1249	616	120	53	275	10	15
21	37	1118	532	113	39	253	7	9
22	43	1657	807	155	68	363	14	25
23	30	994	505	62	42	242	8	13
24	30	986	496	66	42	241	8	13
25	39	1361	646	132	53	322	10	14
26	34	1235	638	76	67	288	10	19
27	37	1123	529	104	52	265	9	14
28	37	1260	626	85	57	276	14	24
29	37	1270	635	86	57	278	14	24
30	43	1779	857	192	80	413	10	20
31	37	1218	593	121	50	282	9	13
32	30	997	500	66	41	247	8	13
33	33	1070	538	63	49	264	9	15
34	21	561	299	21	25	108	8	11
35	39	1424	681	164	60	305	10	15
36	37	1250	619	85	56	275	14	24
37	44	1743	831	196	76	382	11	19
38	37	1262	629	86	57	278	14	23
39	44	1749	832	199	77	383	11	19
40	38	1258	612	117	55	298	10	15

Table 3.

Statement Coverage of the MAL Preprocessor by 60 Operational Useage Cases. (cont.)								
Case	Procs	Exec	Assign	Calls	Do	If	Reads	Writes
41	39	1290	638	101	58	286	12	23
42	36	1351	695	87	71	326	9	18
43	37	1246	619	84	56	275	14	23
44	45	1736	838	172	71	382	15	27
45	45	2002	971	213	86	435	16	28
46	44	1685	819	162	71	371	14	24
47	39	1292	640	101	58	286	12	23
48	45	1683	817	170	70	370	12	23
49	45	1970	956	210	86	417	16	28
50	45	1772	859	178	73	386	15	27
51	39	1337	635	127	54	317	10	16
52	37	1271	635	86	58	280	14	23
53	34	1184	585	109	46	267	9	15
54	40	1355	650	126	52	333	9	13
55	40	1456	689	167	57	327	10	15
56	37	1250	617	120	53	275	10	15
57	37	1248	603	115	54	302	9	14
58	37	1272	637	86	57	278	14	24
59	34	1048	516	72	50	242	9	15
60	20	527	273	19	24	106	8	11
UNION	55	2757	1345	327	120	581	19	36
INTERSECT	19	442	228	16	19	86	7	9
MAXIMUM	68	4300	1870	418	157	753	34	206

Statement Coverage of the MAL Preprocessor by 60 Operational Useage Cases. (Percentage of Maximum)								
Case	Procs	Exec	Assign	Calls	Do	If	Reads	Writes
1	57.4	31.8	35.3	29.9	33.1	43.0	29.4	6.8
2	63.2	39.8	44.5	46.2	51.0	50.6	29.4	9.2
3	66.2	42.6	47.9	44.0	49.7	54.6	38.2	10.7
4	54.4	29.3	33.4	20.6	36.3	36.9	41.2	11.7
5	54.4	29.1	33.0	28.7	33.8	36.7	29.4	7.3
6	52.9	25.5	28.7	20.1	31.8	34.3	26.5	6.8
7	48.5	23.5	26.0	22.5	24.8	31.3	26.5	6.3
8	57.4	31.6	34.9	30.9	33.1	44.0	26.5	6.3
9	54.4	29.0	33.1	20.1	35.7	36.5	41.2	11.2
10	54.4	29.1	33.0	28.7	33.8	36.7	29.4	7.3
11	64.7	40.5	44.4	46.9	48.4	50.7	32.4	9.2
12	54.4	29.0	32.9	28.7	33.8	36.5	29.4	7.3
13	51.5	30.1	35.6	19.4	43.3	40.6	29.4	9.2
14	51.5	29.9	35.3	19.4	43.3	40.5	29.4	9.2
15	51.5	26.4	29.1	25.4	28.7	36.1	26.5	6.8
16	67.6	41.7	45.6	51.9	48.4	50.2	35.3	9.2
17	54.4	29.6	34.1	20.6	36.3	36.9	41.2	11.7
18	54.4	29.1	33.0	28.7	33.8	36.7	29.4	7.3
19	54.4	29.5	34.0	20.6	36.3	36.9	41.2	11.7
20	54.4	29.0	32.9	28.7	33.8	36.5	29.4	7.3
21	54.4	26.0	28.4	27.0	24.8	33.6	20.6	4.4
22	63.2	38.5	43.2	37.1	43.3	48.2	41.2	12.1
23	44.1	23.1	27.0	14.8	26.8	32.1	23.5	6.3
24	44.1	22.9	26.5	15.8	26.8	32.0	23.5	6.3
25	57.4	31.7	34.5	31.6	33.8	42.8	29.4	6.8
26	50.0	28.7	34.1	18.2	42.7	38.2	29.4	9.2
27	54.4	26.1	28.3	24.9	33.1	35.2	26.5	6.8
28	54.4	29.3	33.5	20.3	36.3	36.7	41.2	11.7
29	54.4	29.5	34.0	20.6	36.3	36.9	41.2	11.7
30	63.2	41.4	45.8	45.9	51.0	54.8	29.4	9.7
31	54.4	28.3	31.7	28.9	31.8	37.5	26.5	6.3
32	44.1	23.2	26.7	15.8	26.1	32.8	23.5	6.3
33	48.5	24.9	28.8	15.1	31.2	35.1	26.5	7.3
34	30.9	13.0	16.0	5.0	15.9	14.3	23.5	5.3
35	57.4	33.1	36.4	39.2	38.2	40.5	29.4	7.3
36	54.4	29.1	33.1	20.3	35.7	36.5	41.2	11.7
37	64.7	40.5	44.4	46.9	48.4	50.7	32.4	9.2
38	54.4	29.3	33.6	20.6	36.3	36.9	41.2	11.2
39	64.7	40.7	44.5	47.6	49.0	50.9	32.4	9.2
40	55.9	29.3	32.7	28.0	35.0	39.6	29.4	7.3

Table 4.

Statement Coverage
of the MAL Preprocessor
by 60 Operational Useage Cases.
(Percentage of Maximum)
(cont.)

Case	Procs	Exec	Assign	Calls	Do	If	Reads	Writes
41	57.4	30.0	34.1	24.2	36.9	38.0	35.3	11.2
42	52.9	31.4	37.2	20.8	45.2	43.3	26.5	8.7
43	54.4	29.0	33.1	20.1	35.7	36.5	41.2	11.2
44	66.2	40.4	44.8	41.1	45.2	50.7	44.1	13.1
45	66.2	46.6	51.9	51.0	54.8	57.8	47.1	13.6
46	64.7	39.2	43.8	38.8	45.2	49.3	41.2	11.7
47	57.4	30.0	34.2	24.2	36.9	38.0	35.3	11.2
48	66.2	39.1	43.7	40.7	44.6	49.1	35.3	11.2
49	66.2	45.8	51.1	50.2	54.8	55.4	47.1	13.6
50	66.2	41.2	45.9	42.6	46.5	51.3	44.1	13.1
51	57.4	31.1	34.0	30.4	34.4	42.1	29.4	7.8
52	54.4	29.6	34.0	20.6	36.9	37.2	41.2	11.2
53	50.0	27.5	31.3	26.1	29.3	35.5	26.5	7.3
54	58.8	31.5	34.8	30.1	33.1	44.2	26.5	6.3
55	58.8	33.9	36.8	40.0	36.3	43.4	29.4	7.3
56	54.4	29.1	33.0	28.7	33.8	36.5	29.4	7.3
57	54.4	29.0	32.2	27.5	34.4	40.1	26.5	6.8
58	54.4	29.6	34.1	20.6	36.3	36.9	41.2	11.7
59	50.0	24.4	27.6	17.2	31.8	32.1	26.5	7.3
60	29.4	12.3	14.6	4.5	15.3	14.1	23.5	5.3
UNION	80.9	64.1	71.9	78.2	76.4	77.2	55.9	17.5
INTERSECT	27.9	10.3	12.2	3.8	12.1	11.4	20.6	4.4

Comparison of Statement Coverage of the MAL Preprocessor by 17 Acceptance Test Cases and 60 Operational Usage Cases.								
Case	Procs	Exec	Assign	Calls	Do	If	Reads	Writes
Acpt	51	2408	1187	286	108	490	14	30
Usage	55	2757	1345	327	120	581	19	36
Union	55	2768	1353	327	120	581	19	36
Intersect	51	2397	1179	286	108	490	14	30
A-OpU	0	11	8	0	0	0	0	0
OpU-A	4	360	166	41	12	91	5	6

Table 5.

Comparison of Statement Coverage of the MAL Preprocessor by 17 Acceptance Test Cases and 60 Operational Usage Cases. (by percentage of Maximum)								
Case	Procs	Exec	Assign	Calls	Do	If	Reads	Writes
Acpt	75.0	56.0	63.5	68.4	68.8	65.1	41.2	14.6
Usage	80.9	64.1	71.9	78.2	76.4	77.2	55.9	17.5
Union	80.9	64.4	72.4	78.2	76.4	77.2	55.9	17.5
Intersect	75.0	55.7	63.0	68.4	68.8	65.1	41.2	14.6
A-OpU	0.0	0.3	0.4	0.0	0.0	0.0	0.0	0.0
OpU-A	5.9	8.4	8.9	9.8	7.6	12.1	14.7	2.9

Table 6.

Kruskal-Wallis Comparison of Acceptance Test and Operational Usage Coverage. HO: populations are the same.		
Coverage Type	H	Result
Procedure	0.076	ftr @ 0.1
Executable Statement	0.218	ftr @ 0.1
Assignment	0.076	ftr @ 0.1
Call	0.005	ftr @ 0.1
Do	0.005	ftr @ 0.1
If	0.362	ftr @ 0.1
Read	11.657	reject @ 0.001
Write	2.608	ftr @ 0.1

Table 7.

Mann-Whitney Comparison
of Acceptance Test
and Operational Usage Coverage.

H0: populations are the same.

Coverage Type	U	low U	Result
Procedure	487.5	AT	ftr @ 0.1
Executable Statement	472.0	AT	ftr @ 0.1
Assignment	487.5	AT	ftr @ 0.1
Call	504.0	AT	ftr @ 0.1
Do	504.0	OpU	ftr @ 0.1
If	461.0	AT	ftr @ 0.1
Read	232.0	AT	reject @ 0.01
Write	378.5	AT	ftr @ 0.1

Table 8.

Procedures Classified by the Number of Times Procedure was Exercised / Total Operational Executions (Faulty procedures are starred.) (Unexecuted procedures are u´s)	Procedures
100%	* * * p p p p p p p p p p p p p p p p p p p p p p p p
90%	p p p
80%	* * p p
70%	p
60%	p p p p
50%	* p p p
40%	* p p p p p
30%	p p p
20%	
10%	* p p p p p p
0%	u u u u u u u u u u u u u

Table 9.

Time to Isolate the Change vs
Number of Times Procedure was Exercised /
Total Operational Executions.

(Effort to Understand and Implement in Parenthesis)

100%	(1h < 1d)	(1h < 1d)		
		(1d < 3d)		
90%				
80%		(1h < 1d)	(1d < 3d)	
70%				
60%				
50%	(1h < 1d)			
40%	(1 hour <)			
30%				
20%				
10%		(1h < 1d)		
	< 1 hour	1 hour < 1 day	> 1 day	never found

Table 10.

Time to Understand and Implement the Change vs
Number of Times Procedure was Exercised /
Total Operational Executions.

(Effort to Isolate the Cause in Parenthesis)

	< 1 hour	1 hour < 1 day	1 day < 3 days	> 3 days
100%		(1 h < 1 d) (1 hour <)	(1 h < 1 d)	
90%				
80%		(1 h < 1 d)	(>1 day)	
70%				
60%				
50%		(1 hour <)		
40%	(1 hour <)			
30%				
20%				
10%		(1 h < 1 d)		

Table 11.

Faults by CRF Classification vs
Number of Times Procedure was Exercised /
Total Operational Executions.

	Req.	Func.	Design		Extern.	Lang.	Cler.	Other
		Specs.	Data	Exp	Env.			
100%				x,x		x		
90%								
80%		x	x					
70%								
60%								
50%						x		
40%					x			
30%								
20%								
10%			x					

Table 12.