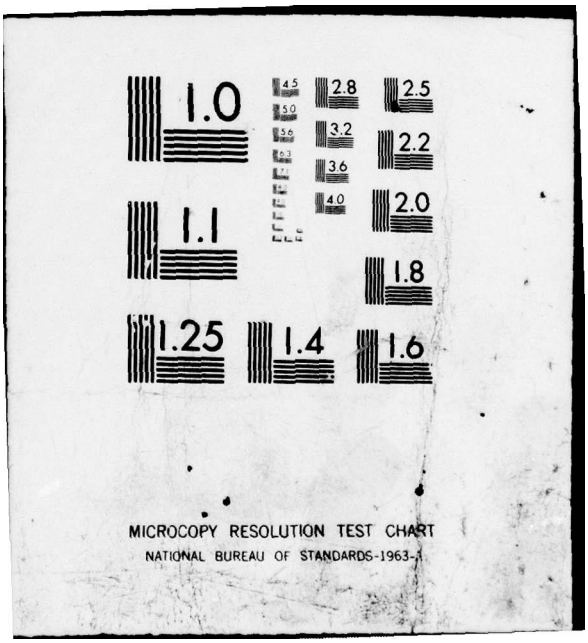
AD-A074 213 ALABAMA UNIV IN HUNTSVILLE SCHOOL OF SCIENCE AND ENG--ETC F/G 9/2
AN ASSESSMENT OF SOFTWARE QUALITY ASSURANCE.(U)
JUL 79 P HSIA, F E PETRY DASG60-78-C-0088

UNCLASSIFIED NL

1 OF 1
AD
A074213

END
DATE FILMED
10-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

AN ASSESSMENT OF SOFTWARE QUALITY ASSURANCE

Final Report - Contract DASG60-78-C-0088

Prepared by:

P. Hsia
F. E. Petry

School of Science and Engineering
The University of Alabama in Huntsville
P. O. Box 1247
Huntsville, Alabama 35807

Prepared for:

U. S. Army Ballistic Missile Defense Systems Command
Post Office Box 1500
Huntsville, Alabama 35807

July 1979

AN ASSESSMENT OF SOFTWARE QUALITY ASSURANCE

Final Report - Contract DASG60-78-C-0088

Prepared by:

P. Hsia
F. E. Petry

School of Science and Engineering
The University of Alabama in Huntsville
P. O. Box 1247
Huntsville, Alabama 35807

Prepared for:

U. S. Army Ballistic Missile Defense Systems Command
Post Office Box 1500
Huntsville, Alabama 35807

July 1979

The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation.

ABSTRACT

This final report describes three tasks which can strengthen the Ballistic Missile Defense Systems Command overall research objectives in software systems development. The three tasks are:

- Preprocessor for Verifiable PASCAL (V-PASCAL)
- Evaluation of Software Quality Laboratory (SQLAB) Tools and Techniques
- Testing Research - A New Approach

V-PASCAL is a language designed by GRC to extend PASCAL to permit experimentation with new language constructs which will aid in the development of BMD software.

These capabilities can be implemented with a preprocessor which accepts an input source code written in V-PASCAL, and generates standard PASCAL for compilation, or which extracts for verification purposes descriptive information about the source text. The feasibility of a preprocessor for V-PASCAL has been investigated, and a conclusion is that a revised design of the GRC produced V-PASCAL preprocessor will suffice for the present.

The purpose of the evaluation of the SQLAB tools and techniques is to determine their limitations and assess their enhancements to software quality. It is intended that the results of the evaluation be used as active feedback to the Advanced Quality Assurance Program. This report details the findings of the capabilities and limitations of the tools and techniques included in the SQLAB.

The research in testing is to establish a new technique in software testing by integrating dynamic testing and symbolic execution to form a system that

may in the future replace the correctness proofs of programs. A novel approach is used to partition the input domain of a program to permit a selection of representative test cases. This approach is to be further studied and extended to a variety of looping structures in programs.

TABLE OF CONTENTS

		Page
	Abstract	i
I.	Introduction	1
II.	V-PASCAL Preprocessor	3
III.	Evaluation of SQLAB Tools and Techniques	6
IV.	New Approach to Software Testing	11
V.	Conclusions and Future Research	14
VI.	References	16

I. INTRODUCTION

Software quality has been an important issue even prior to the first Conference on Software Engineering held in Garmish, Germany in 1968 [1]. Twenty years have passed since then and, however, no significant improvement on the quality of software has been demonstrated. Even the term "software quality" is not well-defined and generally accepted by the research community.

This reported research limits the scope of its investigation to the software production process. Based on a commonly believed principle that software produced with fewer errors during its production will have higher quality at the end, this research deals with the problem of efficiency of static and dynamic tools which facilitate the uncoverings of errors in software production.

The mission of the Ballistic Missile Defense Systems Command (BMDSC) has led to the development of software systems which are notable in their size and complexity. Consequently, it has developed a concentrated effort in software research to obtain results which will improve the quality and reduce the overall time and cost of BMD-related software development [2, 3, 4].

BMDSC has contracted General Research Corporation (GRC) to develop research on and demonstrate the feasibility of automated analytical tools to facilitate software development [5]. The objectives of the GRC contract match the interest of the research team in the Computer Science Department at the University of Alabama in Huntsville. The UAH group is involved in work to develop new approaches for the improvement of software quality, and in several projects concerning software methodologies, tools development, and evaluation [6, 7, 8, 9, 10, 11, 12]. Therefore, a contract was granted

to UAH with an objective to evaluate, enhance, and improve the software quality research by GRC.

Three specific tasks are described in the report:

- Pre-processor for V-PASCAL
- Independent Evaluation of Software Quality Laboratory (SQLAB) Tools and Techniques
- Testing Research - A New Approach

This final report summarizes the effort in the past year related to the three tasks and provides a recommendation of future research in the area of software quality research.

II. V-PASCAL PRE-PROCESSOR

Verifiable PASCAL (V-PASCAL) is a language designed by GRC to extend PASCAL to permit experimentation with new language constructs which will aid in the development of BMD software.

The PASCAL language provides a basic framework for programming structured software. Such features as nested procedure declarations, nested control structures, data typing, and data structure hierarchy with controlled access by scope are important capabilities when writing structured code.

The requirement to produce verifiable software can be met, in part, by improving the readability of PASCAL and by adding features to PASCAL to support automated verification. There are a number of changes to PASCAL which respond to these needs. The changes fall into several categories:

- Improved control structures, which are easier to write and which result in more readable code, such as changes to the existing IF-THEN-ELSE statements, etc.
- Executable assertion statements (ASSERT, ENTRY, EXIT) which may be used to report assertion exceptions during testing and can also be used by a program verifier
- Unit qualifiers for variables which declare the physical units in addition to type declaration, thereby making unit consistency checking possible.

These capabilities can be implemented with a pre-processor which accepts an input source code written in V-PASCAL, and generates standard PASCAL for compilation, or which extracts for verification purposes descriptive information about the source text. The syntax of these extensions to PASCAL can be

found in APPENDIX B of Advanced Software Quality Assurance Research Plan [5].

The major issue involved in this research is programming language design, requiring decisions as to which are the most effective extensions to the language. This is basically an experimental task, and a preprocessor provides the means of carrying out evaluations of constructs on which decisions are based for implementation in the final form of the language. It is clear that the preprocessor will have to host new and different structures continually as the research progresses. As various capabilities are found to be feasible or not, constructs for utilizing them will have to be added or deleted to permit evaluations.

The current preprocessor must be modified to permit the addition/deletion of language constructs. Thus it will be necessary to study the current implementation in order to evaluate it and decide if alternative approaches are needed to maintain the required flexibility and overcome the current problems.

In this contract period, the research team has studied the V-PASCAL preprocessor. The V-PASCAL user's manual and other documents were obtained from GRC. Errors have been seeded into various programs and the V-PASCAL's preprocessors capabilities have been evaluated. The findings are summarized:

(1) The V-PASCAL preprocessor took only "perfect" programs as expected inputs. It used to skip large segments of code in V-PASCAL. If the end of the program were skipped, the code was not passed on to the PASCAL compiler and this occurred when major structural errors were made, such as omission or misspelling of THEN in IFTHEN, or structural errors in declarations, etc. The impact of these deficiencies on actual programming is that it poses serious problems for programmers using the V-PASCAL due to excessive effort required to correct programs.

We recommended three alternatives to solve the problem. One of our recommendations was instrumental in correcting the V-PASCAL preprocessor problem. Currently, the preprocessor has been modified such that upon the failure of scan a special routine is called. As a result, a modified PASCAL program is passed to the PASCAL compiler and hence more information concerning the original V-PASCAL program can be obtained through this process.

(2) The structure of the V-PASCAL preprocessor is not conventional because PASCAL is not a proper subset of the language V-PASCAL. Unlike the relationship between FORTRAN and IFTRAN, there are some language constructs which are legal in PASCAL but not accepted by V-PASCAL preprocessor. Due to these differences, the V-PASCAL preprocessor is designed almost like a compiler with extensive syntax analysis and error recovery, instead of the conventional simple key-word type of translating system.

Because of this, the modification of V-PASCAL preprocessor to accommodate new language constructs is almost like constructing a new preprocessor. Hence, the effort of experimenting with new language constructs is not pursued further.

(3) The current status of the V-PASCAL preprocessor allows programs with errors to be translated into PASCAL programs. The disadvantage of the error recovery problem described in (1) has been eliminated. However, due to its structure and implementation, it does not allow for easy changes. Therefore, experiments with new language constructs cannot be carried out easily using this preprocessor. It is highly desirable to construct a simple key word preprocessor to convert the modified constructs into regular PASCAL statements. This is especially important when the human engineering factor is to be investigated in the programming process.

III. EVALUATION OF SQLAB TOOLS AND TECHNIQUES

The overall intent of the Advanced Quality Assurance (AQA) project is to develop tools and techniques integrated in SQLAB to eliminate software errors in large systems at the early stages of development. The tools developed by GRC can be classified into two categories: verification without redundancy and verification with redundancy. The tools in the first category include set/use checks for variables, infinite loop checks for repetitions, and unreachable code checks for portions of programs. In the second category, there are tools for asserted/actual use checks for variables, unit consistency checks for variables or expressions, and specification/code consistency checks for portions of programs.

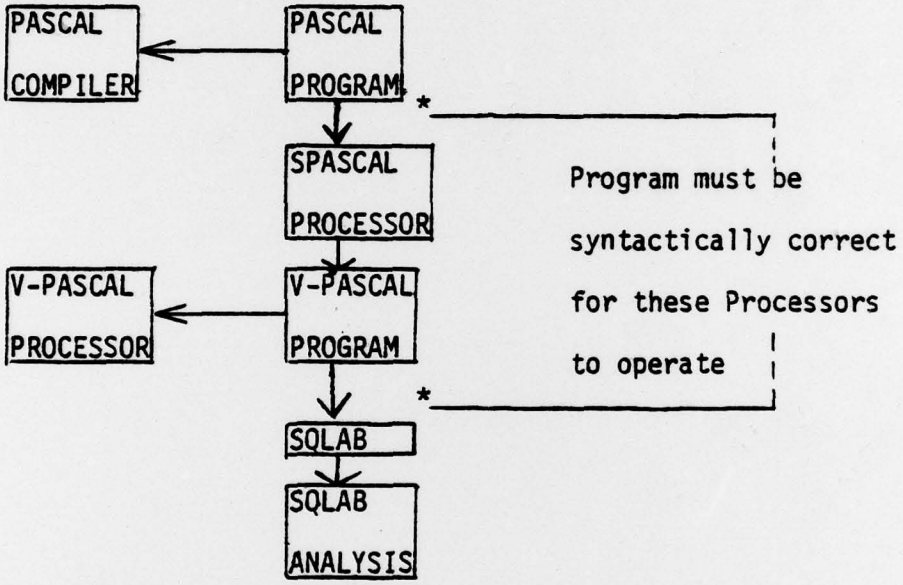
The tools for verification without redundancy can be viewed as passive techniques for programmers because they do not involve direct programmer intervention. The tools for verification with redundancy can be viewed as active techniques since they require the programmer's awareness of the capabilities of the tools and his/her active participation in utilizing them.

In order to determine the limitations of the tools and techniques and their effectiveness in enhancing software quality, two basic studies, local and global effectiveness, were to be carried out. In the first, the tools and techniques will be basically isolated and their individual capabilities examined. Then it is proposed to use a small project to determine the global effectiveness of the interactions of all the individual tools and techniques. Unfortunately, due to the difficulties encountered during the study of the local effectiveness more than the anticipated amount of time has been expended in it. The remaining time was not adequate to carry out a study of the global

effectiveness of the SQLAB. The following two sections are devoted to the descriptions of SQLAB capabilities on V-PASCAL, IFTRAN and FORTRAN.

1. V-PASCAL

As we have been informed by GRC, the current implementation of SQLAB does not provide analysis of PASCAL programs. After discussion with GRC we proposed the following approach to deal with PASCAL programs:



Both SPASCAL and SQLAB require syntactically correct code to proceed. Thus the PASCAL program must first be compiled correctly and likewise the V-PASCAL from SPASCAL (especially if manually re-entered) must not have any V-PASCAL errors which can of course be tested by use of the V-PASCAL processor.

Thus we must rely entirely on the usage of V-PASCAL and so now present the status of SQLAB for V-PASCAL.

A. SQLAB does not handle the record type. As pointed out in Jensen and Wirth, PASCAL User Manual and Report, the record types are perhaps the most flexible of data constructs, being a template for

a structure whose parts have quite distinct characteristics. Thus SQLAB cannot handle this powerful though standard feature of PASCAL.

B. The SQLAB Users Manual is in a draft form and the SQLAB is still changing. On Appendix B of the SQLAB user's manual, May 1978, deck set ups are illustrated for the various aspects of analysis. The examples here have been used for the processing of V-PASCAL, in particular in most cases the set up for static consistency checking. However, many difficulties were encountered in the execution of SQLAB and after consultation with GRC a new deck set up has been used as shown below:

```
            UPDATE (F, N, D, 8)
            ATTACH (X, SQLAB, ID = PGM11)
            X (COMPILE, OUTPUT)
            7/8/9
            *DECK
            (*$U+*)
Program     .
            .
            .
            7/8/9
            LANGUAGE = V-PASCAL
            OPTION CARDS
            7/8/9
            6/7/8/9
```

With this set up most problems seem to have been resolved and testing can proceed again. The continual changes in SQLAB has proven to have been a serious problem for this evaluation. However, this is mainly

due to the research nature of the AQA contract which emphasized the feasibility of the tools instead of the human engineering factors concerning the tools. It is recommended that, for future reference, any research relating to aspects of software engineering should be looked at more intensively from a human factor point of view. This is because the eventual product of software engineering research is to be applied and used by programming personnel. Without a careful look from the human factor point of view, the entire research result may not be used at all.

2. FORTRAN/IFTRAN

SQLAB works well for these two languages. From the beginning of our effort in the evaluation we were advised to the effect that SQLAB tools work for these two languages. Apparently the SQLAB capabilities were first implemented for these two languages. Some features that do not work are the ones that are still in the research stages, such as verification condition generation and loop analysis. However, there are some special programs which can be analyzed with the existing implementation.

The control languages for processing FORTRAN/IFTRAN programs in SQLAB contain less inconsistencies and problems than those for V-PASCAL. But it is strongly recommended that a substantial effort be allotted for the consolidation of the SQLAB user's manual so that a useful tool can be produced.

The following is a table summarizing the current capabilities and status of the SQLAB tools.

STATUS OF SQLAB FUNCTIONS

Language / Function	FORTRAN	IFTRAN	PASCAL	V-PASCAL
LIST	X	X	Not Applicable	X
I/O	X	X		X
UNITS	X	X		X
STATIC	X	X		X
DATA	X	X		X
INSTRUMENTATION	X	X		?
COVERAGE	X	X		?
EXECUTABLE/ASSERTION	X	X		
VCG	Research Stage	Research Stage		Research Stage
LOOP	Research Stage	Research Stage		Research Stage

(RECORDS are not handled by V-PASCAL.)

IV. NEW APPROACH TO SOFTWARE TESTING

The objective of this research is to establish a new approach in software testing by integrating dynamic testing and symbolic execution to form a system that hopefully may replace the correctness proofs of programs in the future.

It has been widely accepted that software testing can only prove the presence, not the absence, of bugs. Therefore, software testing cannot be used to prove the correctness of programs. Although many special techniques have been developed for proving the correctness of programs, none of them are yet either economically feasible or practical enough to be applied to general programs. Consequently, software testing, in spite of its deficiency, remains the only practical means of scrutinizing the quality of software products. The research is an attempt to establish a mathematical foundation for software testing with the hope that finite and manageable test cases can be used to demonstrate the correctness of programs.

Software structure gives rise to the concept of "path." There are two types of paths: structural and execution. Each distinct execution path in a program can be invoked only by a particular set of inputs. Therefore software structure partitions the input domain. Each execution path corresponds to a block of the partition (called path-domain). The structural paths which are not execution paths correspond to null set in the input domain. From the symbolic execution one can obtain a symbolic formula for each execution path. Consequently, a symbolic formula defines the outputs for a certain path-domain in the input domain. With this realization, one does not need to test all inputs from the path-domain to know that the path is correct. Actually any test case from the path-domain may be used to validate the path. Thus, for any software

structure which splits a given input domain into a partition with finite indices one can validate the program by finite number of test cases.

This research uses a novel approach to find out the partition of the input domain by a combination of dynamic testing and symbolic execution. In symbolic execution, one finds it difficult to select test cases by solving systems of inequalities. The new approach does not attempt this generally unsolvable problem. By using dynamic testing with a monitor on predicate nodes, one may use specific test data to find the path-domain that contains the test data by tracing through all the predicate nodes on the path. After the path-domain is determined, its corresponding symbolic formula can also be found as the program is executed. From the symbolic formula, the actual test data, and its dynamic testing result, one will be able to validate the path. In addition, any new test data can be excluded from that equivalence class in the input domain. By applying this criterion of data selection one may eventually exhaust all the path-domains (if the software structure induces a partition with finite index on the input domain). For the input domains with infinite partitions (i.e., partitions with infinite indices) more research is required to enhance this approach.

This research has not yet yielded good results. This is because of the problem of assuring the complete coverage of the entire domain. In simple cases, one may find it easy to demonstrate that the aggregate of the path-domain is equal to the entire input domain. However, no general algorithum has been found to carry out this demonstration effectively. In addition, this problem is closely related to the problem of selecting new test data to find a new region in the input domain for another path. Furthermore, the input domain is in general a set in an n-dimensional space and at the present time, defies any graphical display technique.

It is expected that this new testing approach will work for programs with an input domain in 2-dimensional space. By using a graphics display to describe the input domain and path-domain and also man-machine interaction to find new path-domains, one may eventually demonstrate that the aggregate of all the path-domains completely covers the entire input domain. Thus, the programs can be proved correct in a path-by-path manner.

We recommend that more funding be allocated for this fundamental research since it is an area with potentially high pay-offs.

V. CONCLUSION AND FUTURE RESEARCH

To conclude this report, we want to summarize the achievements of this contract:

(1) Experiences in using the V-PASCAL preprocessor and SQLAB tools have been established.

(2) Timely feedback to GRC to improve the V-PASCAL and SQLAB tools have been provided.

(3) SQLAB tools' capabilities and status have been identified and summarized.

(4) Major problem in the new testing approach has been identified.

With these experiences we recommend that the following list of research be considered for future BMDSC projects to effectively utilize the already accomplished effort and meet the original objective of improving software quality for BMC related software development.

(1) Consolidate the SQLAB tools and techniques.

(2) Produce an up-to-date user's manual for the SQLAB.

(3) Assess the effectiveness of the SQLAB for actual program development.

(4) Construct a preprocessor writing system to enable fast implementation of simple preprocessors. This system can be used to facilitate language construct experimentation. Since it is considerably less complicated than a compiler writing system, it can be designed and constructed with less effort. Because the preprocessors are different from compilers, a different set of design considerations should be used. This system will play a significant role in the human factor research on the future design of programming languages.

(5) Evaluate the design of the SQLAB in terms of portability, modifiability and flexibility in preparation for technology transfer. At the present, it is a

major question as to whether the SQLAB type of capabilities can be implemented on smaller computers (less CDC 7600). It is also unknown as to how many structural changes are needed in SQLAB if ADA, the new DOD standard language, is to be analyzed.

(6) Fundamental research on establishing a theory of testing such as the new testing approach described needs to be supported since it is a potentially high pay-off research.

VI. REFERENCES

1. P. Naur and B. Randall, Software Engineering, NATO Science Committee Report, January 1969.

2. C. G. Davis and C. R. Vick, "The Software Development System," Proceedings of the Second International Conference on Software Engineering, San Francisco, California, October 1976, p. 60.

3. M. Alford, "A Requirements Engineering Methodology for Real-Time Processing Requirements," Proceedings of the Second International Conference on Software Engineering, San Francisco, California, October 1976, pp. 61-68.

4. R. G. Koppang, "Process Design System: An Integrated Set of Software Development Tools," Proceedings of the Second International Conference on Software Engineering, San Francisco, California, October 1976, pp. 86-90.

5. S. Saib, J. Benson, and R. Melton, Advanced Software Quality Assurance Research Plan, CR-4-720, General Research Corporation, November 1976.

6. M. Alford, P. Hsia, and F. Petry, "A Software Engineering Approach to Introductory Programming Courses, SIGCSE Bulletin, Vol. 9, #1, February 1977, pp. 157-162.

7. M. Alford, P. Hsia, and F. Petry, "Use of Software Engineering Methodologies in a Computer Science Curriculum," Proceedings of 15th Annual S.E. Regional ACM Conference, pp. 470-482.

8. M. Alford, P. Hsia, and F. Petry, Reliable Programming in FORTRAN: An Introductory Text, in preparation.

9. P. Hsia and F. Petry. Evaluations of a Disciplined Programming Methodology, submitted for publication.

10. P. Hsia and F. Petry, "Post-processor for Structured FORTRAN," Proceedings of the 14th Annual Southest Regional ACM Conference, Birmingham, Alabama, April 1976, pp. 7-13.

11. M. V. Carby, Post-processor for Structured FORTRAN, M. S. Thesis, Computer Science Program, University of Alabama in Huntsville, February 1978.

12. P. Hsia and F. Petry, Evaluation of PDL-II, Final Report, Contract Number DASG60-77-C-0003, UAH Research Report Number 204, November 1977.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER (9)
4. TITLE (and Subtitle) (6) AN ASSESSMENT OF SOFTWARE QUALITY ASSURANCE		5. TYPE OF REPORT & PERIOD COVERED Final Report, 8 June 78 – 7 July 79
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Pei Hsia, Frederick E. Petry		8. CONTRACT OR GRANT NUMBER(s) (15) DASG60-78-C-0088
9. PERFORMING ORGANIZATION NAME AND ADDRESS The University of Alabama in Huntsville, P. O. Box 1247, Huntsville, AL 35807		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 6.33.04.A
11. CONTROLLING OFFICE NAME AND ADDRESS U S Army Ballistic Missile Defense Systems Command, P. O. Box 1500, Huntsville, AL 35807 (11)		12. REPORT DATE July 1979
		13. NUMBER OF PAGES 16
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 23p.		15. SECURITY CLASS. (of this report) Pending
		15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

~~Pending~~

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software systems, software quality, software tools, pre-processor, programming languages, program correctness, software engineering, dynamic testing, symbolic execution

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Three tasks were performed in this research on software systems development. A pre-processor for Verifiable PASCAL (V-PASCAL) was investigated. A revised design was implemented and this will achieve the overall aims required. The tools and techniques integrated into the Software Quality laboratory were evaluated. It was found that not all features were completed as they were still in the research stage, but the total useable features were delineated and some of the human factors in their use discovered. The

(over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

410243

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

research in testing studied a new technique in software testing which integrates dynamic testing and symbolic execution. A novel approach can be used to partition the input domain of a program to permit a selection of representative test cases.