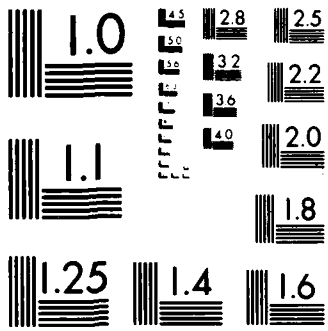
AD-A180 240 METHODOLOGY INVESTIGATION: AUTOMATION OF THE MULTILINGUAL STATIC ANALYSIS TOOL (MSAT)(U) ARMY ELECTRONIC PROVING GROUND FORT HUACHUCA AZ 1/1

UNCLASSIFIED K E VAN KARSEN ET AL. MAR 87 F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

AD NUMBER ______________________

TECOM PROJECT NO. 7-CO-R86-EPO-007

METHODOLOGY INVESTIGATION

FINAL REPORT

AUTOMATION OF THE MULTILINGUAL STATIC ANALYSIS TOOL (MSAT)

BY

K. E. VAN KARSEN

Software and Automation Division
Electronic Surveillance and Security Test Directorate

US ARMY ELECTRONIC PROVING GROUND
FORT HUACHUCA, ARIZONA

MARCH 1987

Prepared for:

US Army Test & Evaluation Command
Aberdeen Proving Ground, MD 21005-5055

DISPOSITION INSTRUCTIONS

Destroy this report in accordance with appropriate regulations when no longer needed. Do not return it to the originator.

DISCLAIMER

Information and data contained in this document are based on input available at the time of preparation. Because the results may be subject to change, this document should not be construed to represent the official position of the U.S. Army Materiel Command unless so stated.

The use of trade names in this report does not constitute an official indorsement or approval of the use of such commercial hardware or software. This report may not be cited for purposes of advertisement.

DEPARTMENT OF THE ARMY
HEADQUARTERS, U.S. ARMY TEST AND EVALUATION COMMAND
ABERDEEN PROVING GROUND, MARYLAND 21005 – 5055

REPLY TO ATTENTION OF

AMSTE-TC-M

SUBJECT: Final Report RDTE Methodology Improvement Program, Multilingual Static Analysis Tool (MSAT) Automation, TECOM Project No. 7-CO-P86-EP3-002

Commander
U.S. Electronic Proving Ground
ATTN: STEEP-MT-DA
Fort Huachuca, Arizona 85613-7110

1. Subject report is approved.

2. TECOM - Providing Soldiers the Decisive Edge.

FOR THE COMMANDER:

Grover H. Shelton

GROVER H. SHELTON
Chief, Methodology Improvement Division
Directorate for Technology

Accesion For		
NTIS CRA&I		☑
DTIC TAB		☐
Unannounced		☐
Justification Report date correct per phonecon		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

QUALITY INSPECTED 4

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
TRMS No. 7-CO-P86-EP3-002

2. GOVT ACCESSION NO.
AD-A180240

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
METHODOLOGY INVESTIGATION FINAL REPORT-AUTOMATION OF MULTI-LINGUAL STATIC ANALYSIS TOOL (MSAT)

5. TYPE OF REPORT & PERIOD COVERED
Final Report

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Edward L. Anderson

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
US Army Electronic Proving Ground
Fort Huachuca, AZ 85613-7110

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
1Y678011DE51

11. CONTROLLING OFFICE NAME AND ADDRESS
US Army Test and Evaluation Command
Attn: AMSTE-TC-M
Aberdeen Proving Ground, MD 21005-5055

12. REPORT DATE
~~September 1986~~ Mar. 87

13. NUMBER OF PAGES
46

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

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 Measurement
Software Metrics
Software Quality
Software Static Analysis Tool
Software Testing
Software Assessment

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

The Multilingual Static Analysis Tool (MSAT) automates the collection and reporting of software design and quality characteristics in a multilingual milieu. The goal of MSAT is to minimize the manual effort associated with the static software assessment of a target software system's design, structure, maintainability, modifications, and conformance with documented design and development standards. This report documents an investigation to enhance the basic capabilities of MSAT. The development of a prototype prologue processor which analyzes target software prologues, and the addition of a multiple level,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) embedded language capability for MSAT are described.

TABLE OF CONTENTS

(BLANK PAGE)

LIST OF FIGURES

(BLANK PAGE)

FOREWORD

Ultrasystems Defense and Space Systems, Incorporated, Sierra Vista, Arizona assisted in the preparation of this document under Contract Number DAEA18-83-C-0003.

(BLANK PAGE)

1. SUMMARY

1.1 Background

MSAT is a software static analysis tool which processes target system software source code written in a variety of computer languages. The rationale for the development of MSAT and design information were documented in a previous methodology investigation report [reference 1, appendix B].

The U.S. Army Electronic Proving Ground (USAEPG) and other Installation/Field Operating Activities (I/FOAs) have been tasked by the U.S. Army Test and Evaluation Command (TECOM) to perform software testing of systems containing embedded computer resources. Comprehensive software testing includes the dynamic testing of performance and reliability (using instrumentation such as the Test Item Stimulator and Hybrid (hardware/software) Monitor developed by USAEPG). It also includes the static testing of software quality (using tools like MSAT). Because complex system functionality in command, control, communications, and intelligence (C^3I) systems is increasingly provided by software, the task of assessing performance and quality features of software is becoming a critical factor in the acquisition process.

Static analysis tools are used to examine the actual target system source code, provide greater visibility of the software design and quantitative measures of software quality as actually implemented. A key by-product is the ability to analyze the correctness of documentation with respect to the implementation. Use of static analysis techniques to support testing of maintainability issues is potentially highly cost-effective since up to 80 percent of software costs are associated with maintenance.

The initial MSAT development produced a software tool to automate the collection and reporting of software design and quality characteristics in a multilingual environment. The goal of MSAT is to minimize the manual effort associated with the static software assessment of a target software system's design, structure, maintainability, modifications, and conformance with documented design and development standards. MSAT consists of: (1) a flexible, language-independent data collection component which extracts and stores items of interest in a data base management system (DBMS); (2) static analysis (SA)/report generation (RG) components for calculating and presenting software metrics and reports; and (3) an executive control component which provides a user-friendly interface.

The current investigation, Automation of the MSAT, was conducted to improve the methods for automating static software analysis and to augment MSAT capabilities where feasible. The investigation included examination of methods to automatically process prologues (commentary at the beginning of software modules) and the demonstration of a prototype prologue processor. The other major area addressed was the provision of multiple levels of computer languages in the development of MSAT.

1.2 Objective

The objective of this investigation was to improve the automation of MSAT. Specific goals were:

- Addition of a prologue processor to automatically identify prologues and provide them to the software analyst.
- Enhancement of the capability to detect and process multiple levels of embedded computer languages.

1.3 Summary of Procedures

Partial funding of the investigation limited the scope of effort, particularly the integration of a prologue processor into MSAT. The level of effort did allow for determining the requirements for a prologue processor and developing a stand-alone prototype version. The Prologue Processor Program (PPP) was then used to demonstrate the capabilities by applying the PPP to several software systems. The results of this work were incorporated into the future design requirements of MSAT. A technique for automatically identifying a specific systems' prologues was then demonstrated by developing a special preprocessor for MSAT prologues.

Remaining effort was used to define and implement the capability for MSAT to detect and process multiple levels of embedded computer languages. Requirements were developed and incorporated into the MSAT specifications. These capabilities were then implemented in the initial version of the software.

1.4 Summary of Results

The automation of MSAT investigation resulted in the development of a table-driven prologue processor tool, the PPP. This tool automatically identifies prologue commentary in software source code, and collects statistical data on the prologue contents. The PPP was applied to software from both inhouse tools and tactical systems. Application of the PPP typically required 2-4 days for experienced personnel to set up the necessary tables and process target software. Performed manually, up to two months have been required to extract similar statistics, with less detail obtained.

The MSAT design provides for the eventual inclusion of a prologue processor. Since funding constraints did not allow integration of the prototype PPP into MSAT, a minimal capability for processing prologues was provided. When identified by the analyst, MSAT will examine and extract the size of prologues, store the collected information in the data base, and report the information on software quality metrics, standards compliance, and change analysis reports. Automatic identification of prologues was demonstrated by developing a special preprocessor which recognized prologues embedded in MSAT software source code.

The effort to provide MSAT with embedded computer language capability resulted in requirement specifications for high order language (HOL), very HOL (VHOL), and assembly language processing. These requirements were satisfied by designing and implementing the capability to process up to three different source languages for each Computer Software Configuration Item (CSCI) component of a target system. Furthermore, each unit of software may contain a mixture of VHOL, HOL, and assembly source code. (In actuality, three different languages can be accommodated regardless of language level. Thus software composed of two different HOLs and a VHOL or assembly language could be processed.)

1.5 Analysis

Application of the PPP demonstrated the feasibility of automatically extracting quantitative metrics from software prologues. Automation of prologue analysis offers a considerable savings in labor compared to manual analysis methods with cost-effectiveness further improved by the more detailed statistics collected by the PPP. Additionally, the extensive effort required for manual analysis is prohibitively expensive for most system tests.

A minimal capability to process prologues was implemented for MSAT. Extraction and reporting functions provide size information. These capabilities would be greatly improved by incorporating a full-scale prologue processor as demonstrated by the prototype PPP. Automatic detection of prologues may be accomplished by system dependent preprocessors, although this requires a high degree of consistency in the format of the target software prologues.

The multiple level embedded language capability of MSAT allows for three levels of any combination of three languages. Usually, embedded languages would be encountered as a combination of VHOL (e.g., DBMS query language) with HOL or HOL with assembly. Since MSAT would allow both of these situations within a single unit, all foreseeable embedded language requirements have been adequately addressed.

1.6 Conclusions

The automation of MSAT effort achieved the objective of the methodology investigation. Development and use of the PPP demonstrated the feasibility of adding such a capability to MSAT. Furthermore, the automatic extraction of prologues was demonstrated by the use of a special purpose preprocessor for MSAT. The multiple level embedded language capability of MSAT meets the requirements anticipated for future test needs. This capability will allow the analysis of software which consists of a variety of languages at various levels.

1.7 Recommendations

The following are recommended for continuing the course of investigation outlined in the methodology proposal.

a. Conduct an investigation to develop a prologue processor capability for MSAT. This would result in an integrated tool which would fully utilize the user-friendly interface, data base storage and retrieval, and report generation features of MSAT. The addition of quantitative metrics for software prologue analysis would make MSAT a more powerful tool.

b. Initiate the remaining tasks, identified in the proposal for this investigation, for enhancement of the basic MSAT capability.

2. DETAILS OF INVESTIGATION

The previous MSAT investigation accomplished the development of a software static analysis tool suited to multilingual test environment. This investigation, the automation of MSAT, showed the feasibility of automating the analysis of software prologues and resulted in a multiple level embedded language analysis capability for MSAT. Complete operational and design features of the PPP and MSAT are contained in the references [references 2 and 3, appendix B].

2.1 Prologue Processor

Software prologues, sometimes referred to as abstracts, consist of commentary usually preceding a software unit. These comments describe various features (e.g., purpose, variables used, inputs, outputs) of the software to assist the maintainer in modifications resulting from enhancements or repairs of defects. Figure 1 is an example of a typical prologue for a small routine.

The quality of prologue commentary directly affects software maintainability. Static analysis of software prologues attempts to ascertain whether a target system's software contains sufficient, accurate, and understandable comments. This analysis includes the measurement of qualitative factors (e.g., understandability) as well as quantitative metrics (e.g., the presence of required items). Performed manually, a thorough analysis of prologue quality may entail a significant amount of effort. For this reason, manual assessment is seldom performed on other than a small subset of the software being tested. Teampack's 457 software routines were subjected to a thorough manual analysis, requiring approximately two months of effort.

2.1.1 Prologue Processing Requirements.

Experience with software development reveals that no standard format exists for software prologues. Even software written in the same computer language may exhibit diverse formats with the extreme case being no prologue at all. This situation is a result of prologues being implemented with essentially free format comment statements, with no formal syntax (beyond the delimiters and rules for comments of a particular language), and consequently no enforcement of standards by compilers. (One could argue that COBOL is an exception to this statement since some leading commentary is provided for in the language description. Although rarely, if ever, employed in mission critical software, this facet of COBOL is mentioned to stimulate thought on establishing required prologue constructs for other languages.)

Although stringent formats for prologues do not exist, (again with one known exception, MIL-STD-1644B, which is, however, now obsolete) an examination of various software development standards and actual software revealed a considerable amount of commonality of required content, if not format. Extracts from software development standards and actual prologues and prologue skeletons are provided in appendix D. An analysis of this information provided a basis for characterizing the types of elements encountered and determining the metrics to be collected.

The model developed to describe prologue requirements for automatic analysis assumes that a target software system contains a set of prologues.

```
#BEGIN
*========================= BEGIN PROLOGUE 1.0 ===============================
*
* NAME:   SCAN
*
* STRUCTURE HID #:   SP.1
*
* PURPOSE:  ADVANCE POINTER TO 1ST DELIMITER CHAR AT OR PAST CURR POSITION
*
* VER #          DATE          PROGRAMMER       ACTION/SDR#
* -----        --------        ----------       -----------
*   1          1 FEB 85        S.LEWIS          CREATED
*===========================================================================
* CALLED BY:  SCANPROC
*
* INPUT:
*  Name:                  Source:            Description:
*  -----                  -------            ------------
*  DELSET                 ARG                INDEX TO DELIM
*  DELIM                  COM SPTABLES       DELIMITER SETS (BOOLEAN ARRAY)
*  BUFFER,PTR             COM SPDATA         BUFFER,CURRENT POSITION
*
* OUTPUT:
*  Name:                  Destination:                Description:
*  -----                  ------------                ------------
*  PTR                    COM SPDATA                  PTR TO 1ST DELIMITER
*                                                     (IN SPECIFIED DELSET)
* CALLS TO:   NONE
*===========================================================================
* INITIAL/ENTRY CONDITIONS:  PTR POSITIONED AT OR BEFORE EOL IN BUFFER
*
* DESIGN/ALGORITHM:
*       IF DELSET > 0 THEN       !OTHERWISE NO SCAN
*           DO WHILE PTR_NOT_AT_DELIMITER
*               PTR=PTR+1
*           END DO                       ! DELSET AND ASCII VALUE OF CHARACTER
*       ENDIF                            ! POINTED TO IN BUFFER ARE USED TO
*                                        ! INDEX CHAR TYPE (DELIMITER=.TRUE.)
*       RETURN
*
* RETURN CONDITION:  PTR POSITIONED AT 1ST DELIMITER CHAR AT OR PAST
*                       INITIAL POSITION
*===========================================================================
* ERRORS: NONE
*
* SPECIAL CHARACTERISTICS: EOL MUST ALLWAYS BE A DELIMITER.
*          ZERO OR NEGATIVE DELSET RESULTS IN NO SCANNING.
*          SCAN WILL NOT ADVANCE PTR IF IT INITIALLY POINTS TO A DELIMITER.
*========================= END PROLOGUE ====================================
#END
```

Figure 1. Prologue from the Prologue Processor Source

Each prologue contains consistently formatted items used to describe certain features of the software source. Associated with each item (e.g., the "purpose" section) is one or more lines of text composed of symbols. It is common practice to identify the text of a particular item by keyword or position with respect to other items. It is equally apparent that extensive use is made of symbols which improve the style and appearance, but which do not contribute otherwise to an understanding of the comments. This material was termed trim; an example is the use of asterisks to box in related text. Other prologue elements which must be considered are non-comment source code embedded within prologues (e.g., data declarations) and material which is best ignored when analyzing prologues (e.g., text which occurs outside the established boundary of a prologue).

A hierarchical list of these prologue elements which influenced the design of the automated prologue processor is as follows:

System (or file). A collection of software source code containing one or more units with prologues and related statements. Software units may be grouped by files, representing higher level components which form the software system.

Prologues. Prologues are contiguous sequences of elements delimited by a prologue beginning/ending pair. Prologues contain items and other symbols.

Items. Items are categories of required or optional information. A particular item may occur any number of times within a given prologue. A prologue is considered to contain a sequence of items delimited by ITEM STARTs.

Lines. A line consists of embedded source code or a comment containing any number of symbols (TEXT, TRIM, or ITEM START) followed by an end-of-line symbol.

Symbols. Symbols are associated with TEXT, TRIM, or ITEM START information. Other symbols delimit lines, consist of special characters (e.g., punctuation), etc.

2.1.2 Prologue Processor Program Development.

The PPP was produced to demonstrate the feasibility of developing an automated static software analysis tool capability for prologues. The main focus of the design was to develop a technique to process prologues and identify the required prologue items. Less emphasis was placed on software to extract prologues, enter syntax tables, and to store or print the output because these areas would change considerably when the tool was integrated with MSAT.

Design of the PPP had to take into consideration the lack of a formal grammar to describe prologues. Successful automatic processing is dependent on the developer of the target software having followed a fairly consistent format in generating prologues. Most static analysis tools have the advantage of processing syntactically correct source code; a prologue processor on the other had is likely to encounter considerable inconsistencies compared to the stringent rules enforced by language compilers.

The result of these considerations was a table-driven tool which can be modified to process a wide range of prologue formats and tolerate the inconsistencies introduced by different programmers. The PPP is essentially a context-sensitive parser with a flexible scanner. It recognizes string patterns (tokens or symbols) based on user defined rules and invokes action routines that accumulate metrics and produce reports. The actions to be taken on encountering a given token can be varied according to the specific user defined state which is active.

To facilitate the application of the PPP, a trace feature was incorporated to provide a detailed history of a run. Information about the input, decision table row, action routine, and token matching is available as an aid to developing new tables. Another feature controlled by the decision table is a reduced output which is essentially a translation of the original prologue symbols into the metasymbols defined by the prologue model (i.e., TEXT, TRIM, etc.).

An example of the statistics which may be collected with the PPP is shown in figure 2. Statistics from processing a group of prologues are summarized by the quantity (lines and symbols) of text found: within items (item-text), outside of recognized items (unassociated text), and total text symbols (prologue-total). Deviations among these metrics can alert the analyst to potential problems in processing or in prologue quality. Of greater importance is the number of items with text (missing items usually are representative of a defect) and items without text (usually caused by a prologue template which is incompletely filled out). Other reports produced by the PPP and detailed explanations of sample processing are provided in the PPP documentation [reference 2, appendix B].

2.1.3 Operational Lessons Learned.

This investigation revealed that automation of the qualitative features of prologues is not practical with current technology, however it is feasible to automate the collection of quantitative metrics. This approach provides statistics on all of the software units of a system. Summary results can be used directly in test reporting; individual unit metrics can be used to identify a sample of units to be examined for qualitative features. Examining the metrics provided by an automated tool also offers the potential of detecting patterns in the implementation. This could lead to the discovery of defects without having to perform an extensive manual analysis.

The PPP was successfully applied to various software systems. The source analyzed included: the PPP itself, Test Item Stimulator, portions of MSAT, Trailblazer, and FORTRAN code analysis tool. Definition and debugging of the decision tables typically required 2-4 days effort. Although additional effort is required to assess qualitative aspects of the target prologues, a considerable improvement in cost-effectiveness is realized over manual methods. The ease in handling idiosyncrasies among the prologues analyzed depends upon the characteristics of the differences and the skill of the user in recognizing common features. This factor should be considered when developing a prologue processor for MSAT. An interactive design should be provided to allow the analyst to compensate for anomalous prologue structures.

PROLOGUE	ITEM-TEXT LINES	ITEM-TEXT SYMBOLS	UNASSOC-TEXT LINES	UNASSOC-TEXT SYMBOLS	PROLOG-TOTAL LINES	PROLOG-TOTAL SYMBOLS	ITEMS W/TEXT	ITEMS W/O TEXT
FOPEN	33	138	2	6	61	186	13	0
LOADIABS	60	282	2	6	96	329	14	0
PPDRIVER	60	303	2	6	93	367	14	0
SCAN ROC	73	436	2	6	106	512	14	0
SCAN	25	138	2	6	48	176	13	0
MATCH	40	174	2	6	63	212	12	0
SYMOUT	41	169	2	6	69	214	13	0
GETLINE	33	166	2	6	61	211	13	0
USER	77	415	2	6	121	480	11	0
ADD2GLOB	17	62	2	6	38	97	10	0
COPY2PRO	17	66	2	6	38	101	10	0
ZEROGLOB	16	50	2	6	37	85	10	0
ZEROPRO	16	50	2	6	37	85	10	0
PREPORT	31	164	2	6	60	210	13	0
****ACCUM TOTALS****	539	2613	28	84	928	3265	170	0

14 PROLOGUES

Figure 2. Summary Report

2.1.4 MSAT Prologue Processor.

Funding constraints prevented the integration of a prologue processor with MSAT. Provision was made for the eventual inclusion of a prologue processor function and a minimal capability was included in the MSAT development. The current MSAT tool will pass manually identified prologue text to a skeleton prologue processor routine. This routine counts the quantity of prologue commentary (excluding blank lines). This information is then stored in a data base for later retrieval and report processing. Prologue text is also stored, and the advanced analyst has the capability to retrieve and review the original material.

Due to variations in the construction of prologues, a general purpose routine to automatically identify and extract prologue text was not developed. Further investigation may result in an algorithm that would be successful in identifying the most common forms of prologues. For special cases, at least, it is feasible to automatically identify prologues. This ability was demonstrated by modifying an MSAT preprocessor to recognize the unique start and end symbols contained in the MSAT source code itself. Although this technique could be used on other target software, a more desirable method would be to develop a general purpose recognizer with externalized rules.

2.2 MSAT Embedded Language Capability

The purpose of MSAT is to automate the collection of software design and quality characteristics from the software source code of a target system. MSAT utilizes a syntax-directed data collection (DC) function to extract information of interest for storage in the MSAT data base (MSDB). Prior to DC processing, a source instrumentation (SI) function preprocesses the target source code to automatically (where possible) identify items such as the beginning and ending of units and internal procedures, and language context switches. Following collection of the data, various SA and RG functions are available for analyzing and reporting on the target system software.

MSAT was developed with multilingual capabilities; the variety of computer languages employed by the systems tested by the USAEPG required a tool with flexible language processing capability. The modular SI design and table-driven DC satisfied this requirement by providing a tool which could be configured for the variation in computer languages among the target systems.

2.2.1 Embedded Language Requirement.

While the original multilingual requirement provided the flexibility to handle different software languages from one application to the next, the proliferation of computers in target systems imposed additional language flexibility requirements. Instead of a single computer with software implemented in one language, multiprocessor systems were being designed, sometimes with a different language used for each processor. Furthermore, with the trend toward use of HOLs, many systems which formerly would have been implemented entirely in assembly language were being developed with a mixture of HOL and assembly.

None of these factors would seriously affect the ability of MSAT to analyze a given system since, as a workaround, each language could be processed as a separate system by MSAT. However, this workaround precludes the generation of summary reports with consolidated information on all of the software in a system. Additional complications would arise from the use of multiple languages within a single software unit. Here the only alternative would be for the analyst to instruct MSAT to "skip" processing of the lines of source code of the secondary language.

Besides assembly language embedded within a HOL, another trend is toward the use of embedded VHOL (e.g., DBMS query language statements). Assuming that HOL is the primary language of implementation, the following combinations of embedded languages are most likely:

1. HOL with separate assembly language units.

2. HOL units with embedded assembly statements.

3. HOL units with embedded VHOL statements.

2.2.2 Implementation of Embedded Language Capability.

The embedded language requirements were identified early enough in the MSAT development to allow changes to incorporate the necessary flexibility in the initial version of the tool. Specifying a multiple language capability within a target system satisfied the first case above, where different languages may appear in separate units of a software component. The second and third cases involved an extension of this capability to allow language context switches within a unit. Fortunately, this capability was allowed for early in the development since the changes required affected every major component to a degree.

The multiple level, multiple language requirements were addressed by arbitrarily assigning a language level (VHOL, HOL, Assembly) for each of up to three languages. The language level associated with each language is arbitrary in the sense that no special processing is performed which is unique to the assigned level. This permits language combinations other than those described above; for example, two HOLs could be accommodated by designating one as a HOL and the other as VHOL (or assembly). Thus any combination of three languages may be analyzed.

The situation of target systems employing multiple processors with multiple languages was provided for by defining target system software as consisting of one or more CSCIs, each of which may use up to three different languages, independent of the languages in other CSCIs. This capability was extended down the software hierarchy to allow single units to contain any mixture of the three languages specified for the CSCI.

APPENDIX A

METHODOLOGY INVESTIGATION PROPOSAL

(BLANK PAGE)

EXHIBIT P-16 (PART I) Date: December 1984

Production Engineering Measures (PEM) Project

RCS DRCMT-835

---

1. Project Number: 216 2035 (TECOM) 2. Fiscal Code: 5197 3. Cost: $350k

4. Project Title: Automation of the Multilingual Static Analysis Tool (MSAT).

5. Name and location of facility/contractor: US Army Electronic Proving Ground, STEEP-MT-DA, Fort Huachuca, AZ 85613-7110.

6. General Objective: Improvement of manufacturing processes, techniques and equipment.

7. Problem: The Army and DOD are continuing to develop, procure and field automated command, control, communications, and intelligence (C^3I) systems. These automated systems utilize different processors and software languages. MSAT has been developed to perform a static analysis of production software. Additional capabilities are required in MSAT to reduce the time and manpower required to add additional software metrics which are currently determined manually.

8. Proposed Solution: This task will improve the necessary methods for automating static software analysis. Additional MSAT capabilities will be specified, designed, tested, and documented to reduce the time and effort to prepare for and perform static software analysis of production system software.

9. Justification: Two key issues in the production of software are software maintainability and configuration management. Both of these tasks are handled most efficiently through the use of automated tools such as MSAT. Preparation to use the MSAT on production software requires up to 3 man-months. The goal of this investigation is to reduce this time by a factor of 2.

10. Benefits:

a. Quantifiable benefits (S/I):_________ Basis: This task will reduce the number of man-hours required to prepare MSAT for use by a factor of two, thus improving production testing capability.

b. Non-quantifiable benefits: This effort facilitates the evaluation of performance test results by exposing the significance of software changes to evaluators with the effect of reducing decision risks.

11. Deliverables:

a. Reports identifying additional automated capabilities required in MSAT.

b. MSAT software in both hardcopy and machine readable form.

c. Updated maintenance and user's manuals.

d. Report of methods and tool verification.

12. Funding Profile and Scheduled Technical Completion Dates:

Fiscal Year	$Costs, (XK)	Month-Year
FY 86	$130K	Sep 86
FY 87	$120K	Sep 87
FY 88	$100K	Sep 88
TOTAL	$350K	

13. End Items Supported:

a. Primary - Effort will support the production testing of software on all automated systems.

b. Secondary - The quality of the software directly affects the performance, maintainability and supportability of all automated system.

14. Key Milestone Dates:

a. PEP Completion for primary end items - N/A

b. MMT Completion - Sep 89

c. Primary End Item TC - N/A

d. Start of Full Scale Production - N/A

e. Preliminary Design Criteria for Facility - N/A

15. Related MMT and Feasibility Demonstration Efforts:

a.	Project Nos.	0855071, subtask 113
b.	Initiation Date	Feb 84
c.	Completion Date	Sep 85

16. Plan for Implementation of Efforts' Results:

a. When - Limited capability FY 86/87; complete capability FY 88.

b. Where - USAEPG

c. How - Further automation of the MSAT will be used to decrease manpower and time requirements.

d. Who - TECOM field activities responsible for testing and evaluating software and software support centers responsible for software development and maintenance.

e. Cost - No additional costs are anticipated.

17. Energy Resource Impact Statement: This investigation does not involve resources beyond those for study, analysis and computer program generation. Therefore, no impact is expected on energy resources.

18. Project Engineer:

a. Name - Richard G. Jacques.

b. Organization - US Army Electronic Proving Ground, STEEP-MT-DA, Fort Huachuca, AZ 85613-7110.

c. Phone Numbers, AUTOVON 879-1870 Commercial (602)-538-1870

EXHIBIT P-16 (PART II) Date: December 1984

Engineering Measures (PEM) Project

RCS DRCMT-835

Project Number: 216 3035 Fiscal Code: 5197

Project Title: Automation of the Multilingual Static Analysis Tool (MSAT)

19. Project Cost Update:

a. Requested at Budget: $350K, Requested at Apportionment: $350k.

20. Scope of Work:

a. FY 86. Add a prologue processor so that prologues are automatically identified and provided to the analyst. Enhance analysis capabilities to detect and process multiple levels of embedded computer language. Update documentation.

b. FY 87. Provide a library of language "table entries" for MSAT so that the definition of table entries for a language/computer application does not have to be made during the preparation of MSAT for usage. Update documentation.

c. FY 88. Automate the remaining software quality metrics and statistical analysis of the resulting data. Update documentation.

21. Time Phasing:

FY	Milestones	Initiation (Month-Year)	Completion (Month-Year)
86	Add prologue processor and embedded language capability.	Oct 85	Sep 86

87	Library of language capabilities.	Oct 86	Sep 87
88	Automation of final software quality metrics.	Oct 87	Sep 88

22. Detailed Cost Summary:

a. Project Costs (use constant dollars)

Cost Type	Government	Contractor GOCO	Contractor Industry	Total
Direct Material				
Engineering Labor	$50K		$225K	$275K
Equipment				
Test & Evaluation (MACI Projects Only)				
Overhead				
Other Factors	20K		40K	60K
Profit or Fee			15K	15K
TOTAL	$70K		$280K	$350K
TOTAL (Inflated Cost)	$70K		$280K	$350K
Percent of total cost	20%		80	100

b. Fiscal Plan:

	Prior Totals	Budget	Future	
Mfg RDTE	FY 84	FY 85	FY __	FY __
Authorized	$110K	$239K		

Project Number 7-CO-R85-EPO-007

PA,A				
Authorized	None			
Obligated				
Expended				

Additional implementation costs:

Costs for pollution abatement or OSHA:

APPENDIX B

REFERENCES

(BLANK PAGE)

REFERENCES

1. Methodology Investigation Final Report Multilingual Static Analysis Tool (MSAT), dated November 1985. TECOM Project No. 7-CO-R85-EPO-007. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613-7110.

2. Technical User's Manual for Prologue Processor Program (PPP), 1 August 1986. Document no. TUM-01-01. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613-7110.

3. Software Product Specification Multilingual Static Analysis Tool (MSAT), dated 24 April 1986. Document no. SPS-02-00. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613-7110.

(BLANK PAGE)

APPENDIX C

ACRONYMS AND ABBREVIATIONS

(BLANK PAGE)

ACRONYMS AND ABBREVIATIONS

ALP........... Automated Language Processing
AMC........... U.S. Army Materiel Command
ASM........... Assembly
C^3I........... Command, Control, Communications, and Intelligence
COBOL......... Common Business Oriented Language
CSCI.......... Computer Software Configuration Item
DARCOM........ U.S. Army Materiel Development and Readiness Command (now AMC)
DBMS.......... Data Base Management System
DC............ Data Collection
DoD........... Department of Defense
FORTRAN....... Formula Translation
HOL........... High-Order Language
I/FOA......... Installation/Field Operating Activity
MSAT.......... Multilingual Static Analysis Tool
MSDB.......... MSAT Data Base
PPP........... Prologue Processor Program
RG .... Report Generation
SA............ Static Analysis
SI............ Source Instrumentation
TECOM......... U.S. Army Test and Evaluation Command
USAEPG........ U.S. Army Electronic Proving Ground
VHOL.......... Very High Order Language

(BLANK PAGE)

APPENDIX D
SAMPLE PROLOGUES/PROLOGUE REQUIREMENTS

(BLANK PAGE)

1.0 Scope. Following (figures 3-7) are examples of prologues from test tools, C^3I systems, and prologue templates from various sources. Extracts from software development standards are included to illustrate the items typically required (see figures 8-11).

(BLANK PAGE)

```
PROGRAM NAME            : CALCULATE ANGLE CPM
MODULE NAME             : DANGLE
MODULE TAG              : ED0201

PROGRAMMER'S NAME       : JEFFREY VINSON
DATE(FIRST CODED)       : 1 AUG 83
DATE(LAST REVISION)     :
     PROGRAMMER         :
     REASON FOR CHANGE:

MODULE FUNCTION:  THIS MODULE CALCULATES AN ANGLE FROM THE SIGNAL STRENGTHS
     OF THE NIR BEACON.  A TWO'S COMPLEMENT IS PERFORMED IF THE ANGLE IS
     NEGATIVE.  IF THE SHUTTER IS CLOSED, A SHUTTER FACTOR (1/4 OF THE ANGLE)
     IS SUBTRACTED DUE TO NON-LINEARARITIES IN THE HARDWARE.

INPUT TO MODULE         : CAMERA (CALLING PARAMETER),
                          CLOSED,
                          SHUTTER,
                          PEAK1 (CALLING PARAMETER),
                          PEAK2 (CALLING PARAMETER)
OUTPUT FROM MODULE      : ANGLE (RETURNED TO CALLING MODULE)
MODULES CALLED          : DIVIDE CPM (UDIVID)
CALLING MODULES         : READ ANGLE CPM (DREADA)
CRITICAL TIMING PATH    : N/A
MISCELLANEOUS           : N/A
ACCURACY REQUIREMENTS   : N/A
ERROR RECOVERY          : N/A
LOCAL VARIABLES         :
```

DATA ITEM	FORMAT	DESCRIPTION
DELTA	ADDRESS	DIFFERENCE OF PEAK VALUES
SHUTTER FACTOR	ADDRESS	1/4 OF THE RESULTANT ANGLE
SUM	ADDRESS	SUM OF PEAK VALUES

Figure 3. Sample C3I System Prologue

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C  TITLE ........... MNEMONIC
C                    LONG DESCRIPTIVE TITLE
C
C  WRITTEN BY ...... AUTHOR NAME  XX/XX/XX
C
C  MODIFIED BY ..... PROGRAMMER   XX/XX/XX   DRF #XXX
C
C  FUNCTION ........ SINGLE SENTENCE FUNCTIONAL OVERVIEW.
C
C  CALLING SEQ ..... CALL MNEMONIC (PAR 1, PAR 2, ..., PAR N)
C
C  CALLS TO ........ LIST ROUTINES CALLED BY THIS MODULE.
C
C  METHOD .......... INCLUDE PDL DESCRIPTION OF THE MODULE'S FUNCTION.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Figure 4. TIS Prologue Skeleton

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C  TITLE ........... EROUT
C                    ERROR OUTPUT ROUTINE
C
C  WRITTEN BY ...... C. HIGHTREE  08/17/81
C
C  MODIFIED BY ..... XXXXXXXXXXX  XX/XX/XX
C
C  FUNCTION ........ DISPLAY FORTRAN SYSTEM ERROR CODES ON THE CRT
C
C  CALLING SEQ ..... CALL EROUT (IERR)
C
C  CALLS TO ........ BASK:  BINARY TO ASC.I CONVERSION
C                    CIDOUT:  CRT DISPLAY ROUTINE
C
C  METHOD .......... DATA
C                      VARIABLE
C                      IERR (INTEGER); ERROR NUMBER; INPUT
C                      ARRAY
C                      MESS (CHARACTER *36); MESSAGE TO BE SENT TO CRT; OUTPUT
C                    END DATA
C                    PROC
C                      BUILD ERROR MESSAGE
C                      OUTPUT MESSAGE TO CRT
C                    END PROC
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      SUBROUTINE EROUT(IERR)
      COMMON/CEROU/MESS(18)
      DATA MESS/FORTRAN SYSTEM ERROR = DECIMAL <15><12>"/
      IPTR=31
      CALL BASK(IERR,4,MESS,IPTR,3,0)
      CALL CIDOUT(MESS,36)
      RETURN
      END
```

Figure 5. Sample TIS Prologue

STANDARD PROLOGUE

```
* * * * * * * * * * * * * * * * * * PROLOGUE * * * * * * * * * * * * * * * * * *
*                                                                               *
** NAME:   Mneumonic - Long Descriptive Title                                   *
*                                                                               *
*                                                                               *
** STRUCTURE ID #:   Number which shows hierarchical position                   *
*                                                                               *
*                                                                               *
** PURPOSE:   Single SENTENCE functional overview                               *
*                                                                               *
*                                                                               *
** CALLING SEQUENCE:   Call Mneumonic (Par 1, Par 2..., ParN)                   *
*                                                                               *
*                                                                               *
** CALLS TO:             List of units (subroutines, etc.) called               *
*                        and 1 line description of their function               *
*                                                                               *
*                     SUBA      READ INPUT VARIABLES                            *
*                                                                               *
*                                                                               *
** INPUTS:            VARIABLE NAME        -        BRIEF DESCRIPTION           *
*                                                                               *
*                                                                               *
*                                                                               *
*                                                                               *
** OUTPUTS:           VARIABLE NAME        -        BRIEF DESCRIPTION           *
*                                                                               *
*                                                                               *
*                                                                               *
*                                                                               *
** FILES:             FILE NAME            -        BRIEF DESCRIPTION & USE     *
*                                                                               *
*                                                                               *
** ALGORITHM:         Description of unit's         Function                    *
*                     (could be PDL for unit -)                                 *
*                                                                               *
** ERRORS:            Description of error conditions which may occur           *
*                     in this unit, error flags, their values and               *
*                     significance.                                             *
*                                                                               *
*                                                                               *
** SPECIAL CHARACTERISTICS:       Any abnormal or special functions              *
*                                 of this program; warnings, notes.             *
*                                                                               *
*                                                                               *
** HISTORY:                                                                     *
*                                                                               *
*   DATE          PROGRAMMER       ACTION          VERSION #        SDR#        *
*                 NAME                             GENERATED                    *
*                                                                               *
*   14 June 84    E. Anderson      Created            1.0           ---         *
*                                                                               *
*                                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Figure 6. MSAT Prologue Template

```
*********************************************************************
* TECOM PROLOGUE                                                    *
*********************************************************************
*                                                                   *
* TITLE                                                             *
*                                                                   *
*-------------------------------------------------------------------*
*                                                                   *
* PURPOSE                                                           *
*                                                                   *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
* INPUTS                                                            *
*                                                                   *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
* OUTPUTS                                                           *
*                                                                   *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
* USAGE                                                             *
*                                                                   *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
* REQUIRED EXTERNALS/FILES                                          *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
* DESCRIPTION                                                       *
*                                                                   *
*                                                                   *
*                                                                   *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
* LIMITATIONS                                                       *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
* AUTHOR(S)                                                         *
*                                                                   *
*-------------------------------------------------------------------*
* DATE                                                              *
*-------------------------------------------------------------------*
* HISTORY/MODIFICATIONS                                             *
*                                                                   *
*                                                                   *
*                                                                   *
*-------------------------------------------------------------------*
```

Figure 7. Proposed TECOM Standard Prologue

30.3.11 Comments. Comments shall be set off from the executable source code in a uniform manner. Before each Unit's executable section, a prologue section shall describe the following details:

a. The Unit's purpose and how it works.

b. Functions, performance requirements, and external interfaces of the CSCI that the Unit helps implement.

c. Other Units (subroutines, procedures, functions) called and the calling sequence.

d. Inputs and outputs, including data files referenced during Unit entry or execution. For each referenced file, the name of the file, usage (input, output, or both), and brief summary of the purpose for referencing the file.

e. Use of global and local variables and, if applicable, registers and memory locations.

f. The identification of special tasks that are internally defined, and the size/structure of which are based on external requirements.

g. The programming department or section responsible for the Unit.

h. Date of creation of the Unit.

i. Date of latest revision, revision number, problem report number and title associated with revision.

Figure 8. Extract from DoD-STD-2167.

5.4.8 Abstracts. Each hierarchical component (i.e., program, subprogram, module, and unit) shall include at the beginning of the executable code a textual description of its inputs, outputs, and function or task; and a list of other components called. In addition to general explanations, to assist understanding, precise references to the appropriate statement labels and data-names shall be included in each module and unit descriptive abstract. The descriptive abstract shall define the allowed and expected range of values for all inputs and shall define the allowed and expected range of values for all outputs. A history of the original and updating programmer names, dates, and reasons for all changes, the activity or commercial company name, and the activity or company division code or billet identifier shall be included. Additionally, the abstract shall include a description of any transportability constraints.

Figure 9. Extract from DoD-STD-1679A

```
*****************************************************************
*M103                    MANAGEMENT INFORMATION                 *
*    DEVICE ID: XXXXXXXX          HID:      XX.XX.XX.XX.XX.XX-XX *
*    FILENAME:  XXXXXXXXXXXX      LANGUAGE: XXXXXXXX            *
*    LOAD MODUL:XXXXXXXXXXXX      SYSTEM:   XXXXXXXX            *
*    COMPILER:  XXXXXXXXXXXX      ASSEMBLER:XXXXXXXXXXXX        *
*    TRNR CPU:  XXXXXXXX          CPU TYPE: XXXXXXXXXXXX        *
*    ITRTN RATE:XXXXXXXX          MODE:     XXXXXXXXXXXX        *
*    LINK EDITR:XXXXXXXXXXXX                                    *
*                                                               *
*****************************************************************
*PU02                           PURPOSE                         *
*    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                                                               *
*****************************************************************
*TI02                    TECHNICAL INFORMATION                  *
*    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                                                               *
*****************************************************************
*PA02                      PROGRAM ATTRIBUTES                   *
*    LOGICAL RECORDS:          XXXXXX  GENERAL COMMENTS:  XXXXXX *
*    FORTRAN LINES:            XXXXXX  NON-FORTRAN LINES: XXXXXX *
*    EXECUTABLE FTN STMTS:     XXXXXX  NON-EXEC FORTRAN STMTS: XXXXXX *
*                                                               *
*****************************************************************
*SU01                       SUBPROGRAM USAGE                    *
*      NAME      NAME      NAME        NAME      NAME      NAME *
*    XXXXXXXX  XXXXXXXX  XXXXXXXX    XXXXXXXX  XXXXXXXX  XXXXXXXX *
*    XXXXXXXX  XXXXXXXX  XXXXXXXX    XXXXXXXX  XXXXXXXX  XXXXXXXX *
*    XXXXXXXX  XXXXXXXX  XXXXXXXX    XXXXXXXX  XXXXXXXX  XXXXXXXX *
*                                                               *
*****************************************************************
*CH02                        CHANGE HISTORY                     *
*    REV   DATE         SOFTWARE ANALYST              EMPLOYER  *
*                          DESCRIPTION OF CHANGE                *
*    XX  DDMMMYY   XXXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*    XX  DDMMMYY   XXXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*    XX  DDMMMYY   XXXXXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX *
*                                                               *
*****************************************************************
```

Figure 10. Prologue Template per MIL-STD-1644B

```
*************************************************************************TVEL0000
*MI03                       MANAGEMENT INFORMATION                      *TVEL0010
*    DEVICE  ID:99X999D                 HID:        EM.02.01.BA-01      *TVEL0020
*    FILENAME:  TVELBA.FA1              LANGUAGE:   FORTRAN             *TVEL0030
*    LOAD MODUL:FLT IND                 SYSTEM:     EOM                 *TVEL0040
*    COMPILER:  VIIO R04-00             ASSEMBLER:  N/A                 *TVEL0050
*    TRNR CPU:  FLIGHT                  CPU TYPE:   P/E 3240            *TVEL0060
*    ITRTN RATE:30 IPS                  MODE:       OP/DE/DY            *TVEL0070
*    LINK EDITR:LINK R00-00                                             *TVEL0080
*                                                                       *TVEL0090
*************************************************************************TVEL0100
*PU02                            PURPOSE                                *TVEL0110
*    THIS ROUTINE COMPUTES THE ANGLE OF ATTACK AND SIDESLIP ANGLE       *TVEL0120
*    TOGETHER WITH THE RATE OF CHANGE OF BOTH AOA AND SIDESLIP          *TVEL0130
*                                                                       *TVEL0140
*************************************************************************TVEL0150
*TI02                       TECHNICAL INFORMATION                       *TVEL0160
*    FILE FLTEXC.FA3 REQUIRED FOR INCLUDE                               *TVEL0170
*    COMPILER OPTIONS:  H X                                             *TVEL0180
*                                                                       *TVEL0190
*************************************************************************TVEL0200
*PA02                         PROGRAM ATTRIBUTES                        *TVEL0210
*    LOGICAL RECORDS:              250   GENERAL COMMENTS:           60 *TVEL0220
*    FORTRAN LINES:                120   NON-FORTRAN LINES:           0 *TVEL0230
*    EXECUTABLE FTN STMTS:          30   NON-EXEC FORTRAN STMTS:     90 *TVEL0240
*                                                                       *TVEL0250
*************************************************************************TVEL0260
*SU01                          SUBPROGRAM USAGE                         *TVEL0270
*     NAME        NAME        NAME        NAME        NAME        NAME  *TVEL0280
*    AEROANG     TVELIA                                                 *TVEL0290
*                                                                       *TWEL0300
*************************************************************************TVEL0310
*CH02                           CHANGE HISTORY                          *TVEL0320
*    REV   DATE          SOFTWARE ANALYST                EMPLOYER       *TVEL0330
*                                DESCRIPTION OF CHANGE                  *TVEL0340
*    00   06NOV80   HARRY D. CODER           MIRACLE TRAINERS, INC.     *TVEL0350
*              INITIAL IMPLEMENTATION OF SOURCE MODULE                  *TVEL0360
*    01   14FEB81   M. Y. VALENTINE          MIRACLE TRAINERS, INC.     *TVEL0370
*              INCORPORATE LIMIT CHECKING ON EXCURSIONS OF INPUT        *TVEL0380
*              VARIABLES.                                               *TVEL0390
*                                                                       *TVEL0400
*************************************************************************TVEL0410
```

Figure 11. Sample Prologue (Header) from MIL-STD-1644B

(BLANK PAGE)

APPENDIX E

DISTRIBUTION

(BLANK PAGE)

DISTRIBUTION LIST

Addressee	Number of Copies
Commander U.S. Army Test and Evaluation Command ATTN: AMSTE-TC-M	3
AMSTE-TO	2
AMSTE-EV-S	1
AMSTE-TE	6
Aberdeen Proving Ground, MD 21005-5055	
Commander Defense Technical Information Center ATTN: DTIC-DDR	2
Cameron Station Alexandria, VA 22314-5000	
Commander U.S. Army Combat Systems Test Activity ATTN: STECS-DA-M	2
Aberdeen Proving Ground, MD 21005-5000	
Commander U.S. Army Yuma Proving Ground ATTN: STEYP-MSA	2
Yuma, AZ 85634-5000	
Commander U.S. Army Jefferson Proving Ground ATTN: STEJP-TD-E	1
Madison, IN 47250-5000	
Commander U.S. Army Dugway Proving Ground ATTN: STEDP-PO-P	1
Dugway, UT 84022-5000	
Commander U.S. Army Cold Regions Test Center ATTN: STECR-TM	1
APO Seattle, WA 98733-5000	
Commander U.S. Army Electronic Proving Ground ATTN: STEEP-TM-AC	4
Fort Huachuca, AZ 85613-7110	
Commander U.S. Army Tropic Test Center ATTN: STETC-TD-AB	1
APO Miami, FL 34004-5000	

Addressee	Number of copies
Commander U.S. Army White Sands Missile Range ATTN: STEWS-TE-PY	4
STEWS-TE-O	1
STEWS-TE-M	1
STEWS-TE-A White Sands Missile Range, NM 88002-5000	1

(BLANK PAGE)

(BLANK PAGE)

END

10-87

DTIC