NCSC-TG-014-89
Library No. S-231,308

FOREWORD

This publication, Guidelines for Formal Verification Systems, is issued by the National Computer Security Center (NCSC) under the authority and in accordance with Department of Defense (DoD) Directive 5215.1, "Computer Security Evaluation Center." The guidelines defined in this document are intended for vendors building formal specification and verification systems that trusted system developers may use in satisfying the requirements of the Department of Defense Trusted Computer System Evaluation Criteria (TCSEC), DoD 5200.28-STD, and the Trusted Network Interpretation of the TCSEC.

As the Director, National Computer Security Center, I invite your recommendations for revision to this technical guideline. Address all proposals for revision through appropriate channels to the:

National Computer Security Center
9800 Savage Road
Fort George G. Meade, MD, 20755-6000,
Attention: Chief, Technical Guidelines Division.

Patrick R. Gallagher, Jr. 1 April 1989
Director
National Computer Security Center

ACKNOWLEDGMENTS

The National Computer Security Center expresses appreciation to Barbara Mayer and Monica McGill Lu as principal authors and project managers of this document. We recognize their contribution to the technical content and presentation of this publication.

We thank the many individuals who contributed to the concept, development, and review of this document. In particular, we acknowledge: Karen Ambrosi, Tom Ambrosi, Terry Benzel, David Gibson, Sandra Goldston, Dale Johnson, Richard Kemmerer, Carl Landwehr, Carol Lane, John McLean, Jonathan Millen, Andrew Moore, Michele Pittelli, Marvin Schaefer, Michael Sebring, Jeffrey Thomas, Tom Vander-Vlis, Alan Whitehurst, James Williams, Kimberly Wilson, and Mark Woodcock. Additionally, we thank the verification system developers and the rest of the verification community who helped develop this document.

TABLE OF CONTENTS

PREFACE

One of the goals of the NCSC is to encourage the development of production-quality verification systems. This guideline was developed as part of the Technical Guideline Program specifically to support this goal.

Although there are manual methodologies for performing formal specification and verification, this guideline addresses verification systems that provide automated support.

Throughout the document, the term developer is used to describe the developer of the verification system. The term vendor is used to describe the individual(s) who are providing support for the tool. These individuals may or may not be the same for a particular tool.

1. INTRODUCTION

The principal goal of the National Computer Security Center (NCSC) is to encourage the widespread availability of trusted computer systems. In support of this goal the DoD Trusted Computer System Evaluation Criteria (TCSEC) was created, against which computer systems could be evaluated. The TCSEC was originally published on 15 August 1983 as CSC-STD-001-83. In December 1985 the DoD modified and adopted the TCSEC as a DoD Standard, DoD 5200.28-STD. [1]

1.1 PURPOSE

This document explains the requirements for formal verification systems that are candidates for the NCSC's Endorsed Tools List (ETL). [5] This document is primarily intended for developers of verification systems to use in the development of production-quality formal verification systems. It explains the requirements and the process used to evaluate formal verification systems submitted to the NCSC for endorsement.

1.2 BACKGROUND

The requirement for NCSC endorsement of verification systems is stated in the TCSEC and the Trusted Network Interpretation of the TCSEC (TNI). [4] The TCSEC and TNI are the standards used for evaluating security controls built into automated information and network systems, respectively. The TCSEC and TNI classify levels of trust for computer and network systems by defining divisions and classes within divisions. Currently, the most trusted class defined is A1, Verified Design, which requires formal design specification and formal verification. As stated in the TCSEC and TNI, "... verification evidence shall be consistent with that provided within the state of the art of the particular Computer Security Center-endorsed formal specification and verification system used." [1]

Guidelines were not available when the NCSC first considered endorsing verification systems. The NCSC based its initial endorsement of verification systems on support and maintenance of the system, acceptance within the verification community, and stability of the system.

1.3 SCOPE

Any verification system that has the capability for formally specifying and verifying the design of a trusted system to meet the TCSEC and TNI A1 Design Specification and Verification requirement can be considered for placement on the ETL. Although verification systems that have capabilities beyond design verification are highly encouraged by the NCSC, this guideline focuses mainly on those aspects of verification systems that are sufficient for the design of candidate A1 systems.

The requirements described in this document are the primary consideration in the endorsement process. They are categorized as either methodology and system specification or implementation and other support factors. Within each category are requirements for features, assurances, and documentation.

The requirements cover those characteristics that can and should exist in current verification technology for production-quality systems. A production-quality verification system is one that is sound, user-friendly, efficient, robust, well documented, maintainable, developed with good software engineering techniques, and available on a variety of hardware. [2] The NCSC's goal is to elevate the current state of verification technology to production quality, while still encouraging the advancement of research in the verification field.

Since the NCSC is limited in resources for both evaluation and support of verification systems, the ETL may reflect this limitation. Verification systems placed on the ETL will either be significant improvements to systems already on the list or will provide a useful approach or capability that the currently endorsed systems lack.

This guideline was written to help in identifying the current needs in verification systems and to encourage future growth of verification technology. The evaluation process is described in the following section. Verification systems will be evaluated against the requirements listed in sections 3 and 4. Section 5 contains a short list of possibilities for next-generation verification systems. It is not an all-encompassing list of features as this would be counterproductive to the goals of research.

2. EVALUATION APPROACH

A formal request for evaluation of a verification system for placement on the ETL shall be submitted in writing directly to:

National Computer Security Center
ATTN: Deputy C
(Verification Committee Chairperson)
9800 Savage Road
Fort George G. Meade, MD 20755-6000

Submitting a verification system does not guarantee NCSC evaluation or placement on the ETL.

The developers shall submit a copy of the verification system to the NCSC along with supporting documentation and tools, test suites, configuration management evidence, and source code. In addition, the system developers shall support the NCSC evaluators. For example, the developers shall be available to answer questions, provide training, and meet with the evaluation team.

There are three cases in which an evaluation can occur: 1) the evaluation of a new verification system being considered for placement on the ETL, 2) the reevaluation of a new version of a system already on the ETL for placement on the ETL (reevaluation for endorsement), and 3) the reevaluation of a system on the ETL being considered for removal from the ETL (reevaluation for removal).

2.1 EVALUATION OF NEW SYSTEMS

To be considered for initial placement on the ETL, the candidate endorsed tool must provide some significant feature or improvement that is not available in any of the currently endorsed tools. If the verification system meets this requirement, the evaluators shall analyze the entire verification system, concentrating on the requirements described in Chapters 3 and 4 of this document. If a requirement is not completely satisfied, but the developer is working toward completion, the relative significance of the requirement shall be measured by the evaluation team. The team shall determine if the deficiency is substantial or detrimental. For example, a poor user interface would not be as significant as the lack of a justification of the methodology. Requirements not completely satisfied shall be identified and documented in the final evaluation report.

Studies or prior evaluations (e.g., the Verification Assessment Study Final Report) [2] performed on the verification system shall be reviewed. Strengths and weaknesses identified in other reports shall be considered when evaluating the verification system.

The following are the major steps leading to an endorsement and ETL listing for a new verification system.

1) The developer submits a request for evaluation to the NCSC Verification Committee Chairperson.

2) The Committee meets to determine whether the verification system provides a significant improvement to systems already on the ETL or provides a useful approach or capability that the existing systems lack.

3) If the result is favorable, an evaluation team is formed and the verification system evaluation begins.

4) Upon completion of the evaluation, a Technical Assessment Report (TAR) is written by the evaluation team.

5) The Committee reviews the TAR and makes recommendations on endorsement.

6) The Committee Chairperson approves or disapproves endorsement.

7) If approved, an ETL entry is issued for the verification system.

8) A TAR is issued for the verification system.

2.2 REEVALUATION FOR ENDORSEMENT

The term reevaluation for endorsement denotes the evaluation of a new version of an endorsed system for placement on the ETL. A significant number of changes or enhancements, as determined by the developer, may warrant a reevaluation for endorsement. The intent of this type of reevaluation is to permit improvements to endorsed versions and advocate state-of-the-art technology on the ETL while maintaining the assurance of the original endorsed version.

A verification system that is already on the ETL maintains assurance of soundness and integrity through configuration management (see Appendix). The documentation of this process is evidence for the reevaluation of the verification system. Reevaluations are based upon an assessment of all evidence and a presentation of this material by the vendor to the NCSC. The NCSC reserves the right to request additional evidence as necessary.

The vendor shall prepare the summary of evidence in the form of a Vendor Report (VR). The vendor may submit the VR to the NCSC at any time after all changes have ended for the version in question. The VR shall relate the reevaluation evidence at a level of detail equivalent to the TAR. The VR shall assert that assurance has been upheld and shall include sufficient commentary to allow an understanding of every change made to the verification system since the endorsed version.

The Committee shall expect the vendor to present a thorough technical overview of changes to the verification system and an analysis of the changes, demonstrating continuity and retention of assurance. The Committee subsequently issues a rec*ommendation to the Committee Chairperson stating that assurance has or has not been maintained by the current verification system since it was endorsed. If the verification system does not sustain its endorsement, the vendor may be given the option for another review by the Committee. The reevaluation cycle ends with an endorsement determination by the Committee Chairperson, and if the determination is favorable, a listing of the new release is added to the ETL, replacing the previously endorsed version; the old version is then archived.

The following are the major steps leading to an endorsement and ETL listing for a revised verification system.

1) The vendor submits the VR and other materials to the NCSC Verification Committee Chairperson.

2) An evaluation team is formed to review the VR.

3) The team adds any additional comments and submits them to the Verification Committee.

4) The vendor defends the VR before the Committee.

5) The Committee makes recommendations on endorsement.

6) The Committee Chairperson approves or disapproves endorsement.

7) If approved, a new ETL entry is issued for the revised verification system.

8) The VR is issued for the revisedverification system.

2.3 REEVALUATION FOR REMOVAL

Once a verification system is endorsed, it shall normally remain on the ETL as long as it is supported and is not replaced by another system. The Committee makes the final decision on removal of a verification system from the ETL. For example, too many bugs, lack of users, elimination of support and maintenance, and unsoundness are all reasons which may warrant removal of a verification system from the ETL. Upon removal, the Committee makes a formal announcement and provides a written justification of their decision.

Systems on the ETL that are removed or replaced shall be archived. Systems developers that have a Memorandum of Agreement (MOA) with the NCSC to use a verification system that is later archived may continue using the system agreed upon in the MOA. Verification evidence from a removed or replaced verification system shall not be accepted in new system evaluations for use in satisfying the A1 Design Specification and Verification requirement.

The following are the major steps leading to the removal of a verification system from the ETL.

1) The Verification Committee questions the endorsement of a verification system on the ETL.

2) An evaluation team is formed and the verification system evaluation begins, focusing on the area in question.

3) Upon completion of the evaluation, a TAR is written by the evaluation team.

4) The Committee reviews the TAR and makes recommendations on removal.

5) The Committee Chairperson approves or disapproves removal.

6) If removed, a new ETL is issued eliminating the verification system in question.

7) A TAR is issued for the verification system under evaluation.

2.4 BETA VERSIONS

Currently, verification systems are not production quality tools and are frequently being enhanced and corrected. The version of a verification system that has been endorsed may not be the newest and most capable version. Modified versions are known as beta tool versions. Beta versions are useful in helping system developers uncover bugs before submitting the verification system for evaluation.

The goal of beta versions is to stabilize the verification system. Users should not assume that any particular beta version will be evaluated or endorsed by the NCSC. If the developer of a trusted system is using a beta version of a formal verification system, specifications and proof evidence shall be submitted to the NCSC which can be completely checked without significant modification using an endorsed tool as stated in the A1 requirement. This can be accomplished by using either the currently endorsed version of a verification system or a previously endorsed version that was agreed upon by the trusted system developer and the developer's evaluation team. Submitted specifications and proof evidence that are not compatible with the endorsed or agreed-upon version of the tool may require substantial modification by the trusted system developer.

3. METHODOLOGY AND SYSTEM SPECIFICATION

The technical factors listed in this Chapter are useful measures of the quality and completeness of a verification system. The factors are divided into four categories: 1) methodology, 2) features, 3) assurance, and 4) documentation. Methodology is the underlying principles and rules of organization of the verification system. Features include the functionality of the verification system. Assurance is the confidence and level of trust that can be placed in the verification system. Documentation consists of a set of manuals and technical papers that fully describe the verification system, its components, application, operation, and maintenance.

These categories extend across each of the components of the verification system. These components minimally consist of the following:

a) a mathematical specification language that allows the user to express correctness conditions,

b) a specification processor that interprets the specification and generates conjectures interpretable by the reasoning mechanism, and

c) a reasoning mechanism that interprets the conjectures generated by the processor and checks the proof or proves that the correctness conditions are satisfied.

3.1 METHODOLOGY

The methodology of the verification system shall consist of a set of propositions used as rules for performing formal verification in that system. This methodology shall have a sound, logical basis. This requirement is a necessary but not sufficient condition for the endorsement of the system.

3.2 FEATURES

3.2.1 Specification Language

a. Language expressiveness.

The specification language shall be sufficiently expressive to support the methodology of the verification system. This ensures that the specification language is powerful and rich enough to support the underlying methodology. For example, if the methodology requires that a specification language be used to model systems as state machines, then the specification language must semantically and syntactically support all of the necessary elements for modeling systems as state machines.

b. Defined constructs.

The specification language shall consist of a set of constructs that are rigorously defined (e.g., in Backus-Naur Form with appropriate semantic definitions). This implies that the language is formally described by a set of rules for correct use.

c. Mnemonics.

The syntax of the specification language shall be clear and concise without obscuring the interpretation of the language constructs. Traditional symbols from mathematics should be employed wherever possible; reasonably mnemonic symbols should be used in other cases. This aids the users in interpreting constructs more readily.

d. Internal uniformity.

The syntax of the specification language shall be internally uniform. This ensures that the rules of the specification language are not contradictory.

e. Overloading.

Each terminal symbol of the specification language's grammar should support one and only one semantic definition insofar as it increases comprehensibility. When it is beneficial to incorporate more than one definition for a symbol or construct, the semantics of the construct shall be clearly defined from the context used. This is necessary to avoid confusion by having one construct with more than one interpretation or more than one construct with the same interpretation. For example, the symbol "+" may be used for both integer and real addition, but it should not be used to denote both integer addition and conjunction.

f. Correctness conditions.

The specification language shall provide the capability to express correctness conditions.

g. Incremental verification.

The methodology shall allow incremental verification. This would allow, for example, a verification of portions of a system specification at a single time. Incremental verification may also include the capability for performing verification of different levels of abstraction. This allows essential elements to be presented in the most abstract level and important details to be presented at successive levels of refinement.

3.2.2 Specification Processing

a. Input.

All of the constructs of the specification language shall be processible by the specification processor(s). This is necessary to convert the specifications to a language or form that is interpretable by the reasoning mechanism.

b. Output.

The output from the processor(s) shall be interpretable by the reasoning mechanism. Conjectures derived from the correctness conditions shall be generated. The output shall also report errors in specification processing to the user in an easily interpretable manner.

3.2.3 Reasoning Mechanism

a. Compatibility of components.

The reasoning mechanism shall be compatible with the other components of the verification system to ensure that the mechanism is capable of determining the validity of conjectures produced by other components of the verification system. For example, if conjectures are generated by the specification processor that must be proven, then the reasoning mechanism must be able to interpret these conjectures correctly.

b. Compatibility of constructs.

The well-formed formulas in the specification language that may also be input either directly or indirectly into the reasoning mechanism using the language(s) of the reasoning mechanism shall be mappable to ensure compatibility of components. For example, if a lemma can be defined in the specification language as "LEMMA <well-formed formula>" and can also be defined in the reasoning mechanism, then the construct for the lemma in the reasoning mechanism shall be in the same form.

c. Documentation.

The reasoning mechanism shall document the steps it takes to develop the proof. Documentation provides users with a stable, standard reasoning mechanism that facilitates debugging and demonstrates completed proofs. If the reasoning mechanism is defined to use more than one method of reasoning, then it should clearly state which method is used and remain consistent within each method of reasoning.

d. Reprocessing.

The reasoning mechanism shall provide a means for reprocessing completed proof sessions. This is to ensure that users have a means of reprocessing theorems without reconstructing the proof process. This mechanism shall also save the users from reentering voluminous input to the reasoning mechanism. For example, reprocessing may be accomplished by the generation of command files that can be invoked to recreate the proof session.

e. Validation.

The methodology shall provide a means for validating proof sessions independently of the reasoning mechanism. Proof strategies checked by an independent, trustworthy proof checker shall ensure that only sound proof steps were employed in the proof process. Trustworthy implies that there is assurance that the proof checker accepts only valid proofs. The validation process shall not be circumventable and shall always be invoked for each completed proof session.

f. Reusability.

The reasoning mechanism shall facilitate the use of system- and user-supplied databases of reusable definitions and theorems. This provides a foundation for proof sessions that will save the user time and resources in reproving similar theorems and lemmas.

g. Proof dependencies.

The reasoning mechanism shall track the status of the use and reuse of theorems throughout all phases of development. Proof dependencies shall be identified and maintained so that if modifications are made to a specification (and indirectly to any related conjectures/theorems), the minimal set of theorems and lemmas that are dependent on the modified proofs will need to be reproved.

3.3 ASSURANCE, SOUNDNESS, AND ROBUSTNESS

a. Sound basis.

Each of the verification system's tools and services shall support the methodology. This ensures that users can understand the functionality of the verification system with respect to the methodology and that the methodology is supported by the components of the verification system.

b. Correctness.

The verification system shall be rigorously tested to provide assurance that the majority of the system is free of error.

c. Predictability.

The verification system shall behave predictably. Consistent results are needed for the users to interpret the results homogeneously. This will ensure faster and easier interpretation and fewer errors in interpretation.

d. Previous use.

The verification system shall have a history of use to establish stability, usefulness, and credibility. This history shall contain documentation of applications (for example, applications from academia or the developers). These applications shall test the verification system, so that strengths and weaknesses may be uncovered.

e. Error recovery.

The verification system shall gracefully recover from internal software errors. This error handling is necessary to ensure that errors in the verification system do not cause damage to a user session.

f. Software engineering.

The verification system shall be implemented using documented software engineering practices. The software shall be internally structured into well-defined, independent modules for ease of maintainability and configuration management.

g. Logical theory.

All logical theories used in the verification system shall be sound. If more than one logical theory is used in the verification system, then there shall be evidence that the theories work together via a metalogic. This provides the users with a sound method of interaction among the theories.

h. Machine independence.

The functioning of the methodology and the language of the verification system shall be machine independent. This is to ensure that the functioning of the theory, specification language, reasoning mechanism and other essential features does not change from one machine to another. Additionally, the responses that the user receives from each of the components of the verification system should be consistent across the different hardware environments that support the verification system.

3.4 DOCUMENTATION

a. Informal justification.

An informal justification of the methodology behind the verification system shall be provided. All parts of the methodology must be fully documented to serve as a medium for validating the accuracy of the stated implementation of the verification system. The logical theory used in the verification system shall be documented. If more than one logical theory exists in the system, the metalogic employed in the system shall be explained and fully documented. This documentation is essential for the evaluators and will aid the users in understanding the methodology.

b. Formal definition.

A formal definition (e.g., denotational semantics) of the specification language(s) shall be provided. A formal definition shall include a clear semantic definition of the expressions supported by the specification language and a concise description of the syntax of all specification language constructs. This is essential for the evaluators and will aid the users in understanding the specification language.

c. Explanation of methodology.

A description of how to use the methodology, its tools, its limitations, and the kinds of properties that it can verify shall be provided. This is essential for users to be able to understand the methodology and to use the verification system effectively.

4. IMPLEMENTATION AND OTHER SUPPORT FACTORS

The NCSC considers the support factors listed in this section to be measures of the usefulness, understandability, and maintainability of the verification system. The support factors are divided into the following three categories: 1) features, 2) assurances, and 3) documentation.

Two features that provide support for the user are the interface and the base hardware of the verification system. Configuration management, testing, and mainte*nance are three means of providing assurance. (The Appendix contains additional information on configuration management.) Documentation consists of a set of manuals and technical papers that fully describe the verification system, its components, application, operation, and maintenance.

4.1 FEATURES

4.1.1 User Interface

a. Ease of use.

The interface for the verification system shall be user-friendly. Input must be understandable, output must be informative, and the entire interface must support the users' goals.

b. Understandable input.

Input shall be distinct and concise for each language construct and adequately represent what the system requires for the construct.

c. Understandable output.

Output from the components of the verification system shall be easily interpretable, precise, and consistent. This is to ensure that users are provided with understandable and helpful information.

d. Compatibility.

Output from the screen, the processor, and the reasoning mechanism shall be compatible with their respective input, where appropriate. It is reasonable for a specification processor (reasoning mechanism) to put assertions into a canonical form, but canonical forms should be compatible in the specification language (reasoning mechanism).

e. Functionality.

The interface shall support the tasks required by the user to exercise the verification system effectively. This is to ensure that all commands necessary to utilize the components of the methodology are available and functioning according to accompanying documentation.

f. Error reporting.

The verification system shall detect, report, and recover from errors in a specification. Error reporting shall remain consistent by having the same error message generated each time the error identified in the message is encountered. The output must be informative and interpretable by the users.

4.1.2 Hardware Support

a. Availability.

The verification system shall be available on commonly used computer systems. This will help ensure that users need not purchase expensive or outdated machines, or software packages to run the verification system.

b. Efficiency.

Processing efficiency and memory usage shall be reasonable for specifications of substantial size. This ensures that users are able to process simple (no complex constructs), short (no more than two or three pages) specifications in a reasonable amount of time (a few minutes). The processing time of larger, more complex specifications shall be proportional to the processing time of smaller, less complex specifications. Users should not need to wait an unacceptable amount of time for feedback.

4.2 ASSURANCE

4.2.1 Configuration Management

a. Life-cycle maintenance.

Configuration management tools and procedures shall be used to track changes (both bug fixes and new features) to the verification system from initial concept to final implementation. This provides both the system maintainers and the evaluators with a method of tracking the numerous changes made to the verification system to ensure that only sound changes are implemented.

b. Configuration items.

Identification of Configuration Items (CIs) shall begin early in the design stage. CIs are readily established on a logical basis at this time. The configuration management process shall allow for the possibility that system changes will convert non-CI components into CIs.

c. Configuration management tools.

Tools shall exist for comparing a newly generated version with the previous version. These tools shall confirm that a) only the intended changes have been made in the code that will actually be used as the new version of the verification system, and b) no additional changes have been inserted into the verification system that were not intended by the system developer. The tools used to perform these functions shall also be under strict configuration control.

d. Configuration control.

Configuration control shall cover a broad range of items including software, documentation, design data, source code, the running version of the object code, and tests. Configuration control shall begin in the earliest stages of design and development and extend over the full life of the CIs. It involves not only the approval of changes and their implementation but also the updating of all related material to reflect each change. For example, often a change to one area of a verification system may necessitate a change to another area. It is not acceptable to write or update documentation only for new code or newly modified code, rather than for all parts of the verification system affected by the addition or change. Changes to all CIs shall be subject to review and approval.

The configuration control process begins with the documentation of a change request. This change request should include justification for the proposed change, all of the affected items and documents, and the proposed solution. The change request shall be recorded in order to provide a way of tracking all proposed system changes and to ensure against duplicate change requests being processed.

e. Configuration accounting.

Configuration accounting shall yield information that can be used to answer the following questions: What source code changes were made on a given date? Was a given change absolutely necessary? Why or why not? What were all the changes in a given CI between releases N and N+1? By whom were they made, and why? What other modifications were required by the changes to this CI? Were modifications required in the test set or documentation to accommodate any of these changes? What were all the changes made to support a given change request?

f. Configuration auditing.

A configuration auditor shall be able to trace a system change from start to finish. The auditor shall check that only approved changes have been implemented, and that all tests and documentation have been updated concurrently with each implementation to reflect the current status of the system.

g. Configuration control board.

The vendor's Configuration Control Board (CCB) shall be responsible for approving and disapproving change requests, prioritizing approved modifications, and verifying that changes are properly incorporated. The members of the CCB shall interact periodically to discuss configuration management topics such as proposed changes, configuration status accounting reports, and other topics that may be of interest to the different areas of the system development.

4.2.2 Support and Maintenance

The verification system shall have ongoing support and maintenance from the developers or another qualified vendor. Skilled maintainers are necessary to make changes to the verification system.

4.2.3 Testing

a. Functional tests.

Functional tests shall be conducted to demonstrate that the verification system operates as advertised. These tests shall be maintained over the life cycle of the verification system. This ensures that a test suite is available for use on all versions of the verification system. The test suite shall be enhanced as software errors are identified to demonstrate the elimination of the errors in subsequent versions. Tests shall be done at the module level to demonstrate compliance with design documentation and at the system level to demonstrate that software accurately generates assertions, correctly implements the logic, and correctly responds to user commands.

b. Stress testing.

The system shall undergo stress testing by the evaluation team to test the limits of and to attempt to generate contradictions in the specification language, the reasoning mechanism, and large specifications.

4.3 DOCUMENTATION

a. Configuration management plan.

A configuration management plan and supporting evidence assuring a consistent mapping of documentation and tools shall be provided for the evaluation. This provides the evaluators with evidence that compatibility exists between the components of the verification system and its documentation. The plan shall include the following:

1. The configuration management plan shall describe what is to be done to implement configuration management in the verification system. It shall define the roles and responsibilities of designers, developers,

management, the Configuration Control Board, and all of the personnel involved with any part of the life cycle of the verification system.

2. Tools used for configuration management shall be documented in the configuration management plan. The forms used for change control, conventions for labeling configuration items, etc., shall be contained in the configuration management plan along with a description of each.

3. The plan shall describe procedures for how the design and implementation of changes are proposed, evaluated, coordinated, and approved or disapproved. The configuration management plan shall also include the steps to ensure that only those approved changes are actually included and that the changes are included in all of the necessary areas.

4. The configuration management plan shall describe how changes are made to the plan itself and how emergency procedures are handled. It should describe the procedures for performing time-sensitive changes without going through a full review process. These procedures shall define the steps for retroactively implementing configuration management after the emergency change has been completed.

b. Configuration management evidence.

Documentation of the configuration management activities shall be provided to the evaluators. This ensures that the policies of the configuration management plan have been followed.

c. Source code.

Well-documented source code for the verification system, as well as documentation to aid in analysis of the code during the evaluation, shall be provided. This provides the evaluators with evidence that good software engineering practices and configuration management procedures were used in the implementation of the verification system.

d. Test documentation.

Documentation of test suites and test procedures used to check functionality of the system shall be provided. This provides an explanation to the evaluators of each test case, the testing methodology, test results, and procedures for using the tests.

e. User's guide.

An accurate and complete user's guide containing all available commands and their usage shall be provided in a tutorial format. The user's guide shall contain worked examples. This is necessary to guide the users in the use of the verification system.

f. Reference manuals.

A reference manual that contains instructions, error messages, and examples of how to use the system shall be provided. This provides the users with a guide for problem-solving techniques as well as answers to questions that may arise while using the verification system.

g. Facilities manual.

A description of the major components of the software and their interfacing shall be provided. This will provide users with a limited knowledge of the hardware base required to configure and use the verification system.

h. Vendor report.

A report written by the vendor during a reevaluation that provides a complete description of the verification system and changes made since the initial evaluation shall be provided. This report, along with configuration management documentation, provides the evaluators with evidence that soundness of the system has not been jeopardized.

i. Significant worked examples.

Significant worked examples shall be provided which demonstrate the strengths, weaknesses, and limitations of the verification system. These examples shall reflect portions of computing systems. They may reside in the user's guide, the reference manual, or a separate document.

5. FUTURE DIRECTIONS

The purpose of this section is to list possible features for future or beyond-A1 verification systems. Additionally, it contains possibilities for future research -- areas that researchers may choose to investigate. Research and development of new verification systems or investigating areas not included in this list is also encouraged. Note that the order in which these items appear has no bearing on their relative importance.

a. The specification language should permit flexibility in approaches to specification.

b. The specification language should allow the expression of properties involving liveness, concurrency, and eventuality.

c. The reasoning mechanism should include a method for reasoning about information flows.

d. The design and code of the verification system should be formally verified.

e. The theory should support rapid prototyping. Rapid prototyping supports a way of developing a first, quick version of a specification. The prototype provides immediate feedback to the user.

f. The verification system should make use of standard (or reusable) components where possible (for example, use of a standard windowing system, use of a standard language-independent parser, editor, or printer, use of a standard database support system, etc.).

g. The verification system should provide a language-specific verifier for a commonly used systems programming language.

h. The verification system should provide a method for mapping a top-level specification to verified source code.

i. The verification system should provide a tool for automatic test data generation of the design specification.

j. The verification system should provide a means of identifying which paths in the source code of the verification system are tested by a test suite.

k. The verification system should provide a facility for high-level debugging/tracing of unsuccessful proofs.

l. A formal justification of the methodology behind the verification system should be provided.

APPENDIX

CONFIGURATION MANAGEMENT

The purpose of configuration management is to ensure that changes made to verification systems take place in an identifiable and controlled environment. Configuration managers take responsibility that additions, deletions, or changes made to the verification system do not jeopardize its ability to satisfy the requirements in Chapters 3 and 4. Therefore, configuration management is vital to maintaining the endorsement of a verification system.

Additional information on configuration management can be found in A Guide to Understanding Configuration Management in Trusted Systems. [3]

OVERVIEW OF CONFIGURATION MANAGEMENT

Configuration management is a discipline applying technical and administrative direction to: 1) identify and document the functional and physical characteristics of each configuration item for the system; 2) manage all changes to these characteristics; and 3) record and report the status of change processing and implementation. Configuration management involves process monitoring, version control, information capture, quality control, bookkeeping, and an organizational framework to support these activities. The configuration being managed is the verification system plus all tools and documentation related to the configuration process.

Four major aspects of configuration management are configuration identification, configuration control, configuration status accounting, and configuration auditing.

CONFIGURATION IDENTIFICATION

Configuration management entails decomposing the verification system into identifiable, understandable, manageable, trackable units known as Configuration Items (CIs). A CI is a uniquely identifiable subset of the system that represents the small*est portion to be subject to independent configuration control procedures. The decomposition process of a verification system into CIs is called configuration identification.

CIs can vary widely in size, type, and complexity. Although there are no hard-and-fast rules for decomposition, the granularity of CIs can have great practical importance. A favorable strategy is to designate relatively large CIs for elements that are not expected to change over the life of the system, and small CIs for elements likely to change more frequently.

CONFIGURATION CONTROL

Configuration control is a means of assuring that system changes are approved before being implemented, only the proposed and approved changes are implemented, and the implementation is complete and accurate. This involves strict procedures for proposing, monitoring, and approving system changes and their implementation. Configuration control entails central direction of the change process by personnel who coordinate analytical tasks, approve system changes, review the implementation of changes, and supervise other tasks such as documentation.

CONFIGURATION ACCOUNTING

Configuration accounting documents the status of configuration control activities and in general provides the information needed to manage a configuration effectively. It allows managers to trace system changes and establish the history of any developmental problems and associated fixes. Configuration accounting also tracks the status of current changes as they move through the configuration control process. Configuration accounting establishes the granularity of recorded information and thus shapes the accuracy and usefulness of the audit function.

The accounting function must be able to locate all possible versions of a CI and all of the incremental changes involved, thereby deriving the status of that CI at any specific time. The associated records must include commentary about the reason for each change and its major implications for the verification system.

CONFIGURATION AUDIT

Configuration audit is the quality assurance component of configuration management. It involves periodic checks to determine the consistency and completeness of accounting information and to verify that all configuration management policies are being followed. A vendor's configuration management program must be able to sustain a complete configuration audit by an NCSC review team.

CONFIGURATION MANAGEMENT PLAN

Strict adherence to a comprehensive configuration management plan is one of the most important requirements for successful configuration management. The configuration management plan is the vendor's document tailored to the company's practices and personnel. The plan accurately describes what the vendor is doing to the system at each moment and what evidence is being recorded.

CONFIGURATION CONTROL BOARD

All analytical and design tasks are conducted under the direction of the vendor's corporate entity called the Configuration Control Board (CCB). The CCB is headed by a chairperson who is responsible for assuring that changes made do not jeopardize the soundness of the verification system. The Chairperson assures that the changes made are approved, tested, documented, and implemented correctly.

The members of the CCB should interact periodically, either through formal meetings or other available means, to discuss configuration management topics such as proposed changes, configuration status accounting reports, and other topics that may be of interest to the different areas of the system development. These interactions should be held to keep the entire system team updated on all advancements or alterations in the verification system.

GLOSSARY

Beta Version

Beta versions are intermediate releases of a product to be tested at one or more customer sites by the software end-user. The customer describes in detail any problems encountered during testing to the developer, who makes the appropriate modifications. Beta versions are not endorsed by the NCSC, but are primarily used for debugging and testing prior to submission for endorsement.

Complete

A theory is complete if and only if every sentence of its language is either provable or refutable.

Concurrency

Simultaneous or parallel processing of events.

Configuration Accounting

The recording and reporting of configuration item descriptions and all departures from the baseline during design and production.

Configuration Audit

An independent review of computer software for the purpose of assessing compliance with established requirements, standards, and baselines. [3]

Configuration Control

The process of controlling modifications to the system's design, hardware, firmware, software, and documentation which provides sufficient assurance that the system is protected against the introduction of improper modification prior to, during, and after system implementation. [3]

Configuration Control Board (CCB)

An established vendor committee that is the final authority on all proposed changes to the verification system.

Configuration Identification

The identifying of the system configuration throughout the design, development, test, and production tasks. [3]

Configuration Item (CI)

The smallest component tracked by the configuration management system. [3]

Configuration Management

The process of controlling modifications to a verification system, including documentation, that provides sufficient assurance that the system is protected against the introduction of improper modification before, during, and after system implementation.

Conjecture

A general conclusion proposed to be proved upon the basis of certain given premises or assumptions.

Consistency (Mathematical)

A logical theory is consistent if it contains no formula such that the formula and its negation are provable theorems.

Consistency (Methodological)

Steadfast adherence to the same principles, course, form, etc.

Correctness

Free from errors; conforming to fact or truth.

Correctness Conditions

Conjectures that formalize the rules, security policies, models, or other critical requirements on a system.

Design Verification

A demonstration that a formal specification of a software system satisfies the correctness conditions (critical requirements specification).

Documentation

A set of manuals and technical papers that fully describe the verification system, its components, application, and operation.

Endorsed Tools List (ETL)

A list composed of those verification systems currently recommended by the NCSC for use in developing highly trusted systems.

Eventuality

The ability to prove that at some time in the future, a particular event will occur.

Formal Justification

Mathematically precise evidence that the methodology of the verification system is sound.

Formal Verification

The process of using formal proofs to demonstrate the consistency (design verification) between a formal specification of a system and a formal security policy model or (implementation verification) between the formal specification and its program implementation. [1]

Implementation Verification

A demonstration that a program implementation satisfies a formal specification of a system.

Informal Justification

An English description of the tools of a verification system and how they interact. This includes a justification of the soundness of the theory.

Language

A set of symbols and rules of syntax regulating the relationship between the symbols, used to convey information.

Liveness

Formalizations that ensure that a system does something that it should do.

Metalogic

A type of logic used to describe another type of logic or a combination of different types of logic.

Methodology

The underlying principles and rules of organization of a verification system.

Production Quality Verification System

A verification system that is sound, user-friendly, efficient, robust, well-documented, maintainable, well-engineered (developed with software engineering techniques), available on a variety of hardware, and promoted (has education available for users). [2]

Proof

A syntactic analysis performed to validate the truth of an assertion relative to an (assumed) base of assertions.

Proof Checker

A tool that 1) accepts as input an assertion (called a conjecture), a set of assertions (called assumptions), and a proof; 2) terminates and outputs either success or failure; and 3) if it succeeds, then the conjecture is a valid consequence of the assumptions.

Reasoning Mechanism

A tool (interactive or fully automated) capable of checking or constructing proofs.

Safety Properties

Formalizations that ensure that a system does not do something that it should not do.

Semantics

A set of rules for interpreting the symbols and well-formed formulae of a language.

Sound

An argument is sound if all of its propositions are true and its argument form is valid. A proof system is sound relative to a given semantics if every conjecture that can be proved is a valid consequence of the assumptions used in the proof.

Specification Language

A logically precise language used to describe the structure or behavior of a system to be verified.

Specification Processor

A software tool capable of receiving input, parsing it, generating conjectures (candidate theorems), and supplying results for further analysis (e.g., reasoning mechanism).

Syntax

A set of rules for constructing sequences of symbols from the primitive symbols of a language.

Technical Assessment Report (TAR)

A report that is written by an evaluation team during an evaluation of a verification system and available upon completion.

Theorem

In a given logical system, a well-formed formula that is proven in that system.

Theory

A formal theory is a coherent group of general propositions used as principles of explanation for a particular class of phenomena.

User-Friendly

A system is user-friendly if it facilitates learning and usage in an efficient manner.

Valid

An argument is valid when the conclusion is a valid consequence of the assumptions used in the argument.

Vendor Report (VR)

A report that is written by a vendor during and available upon completion of a reevaluation of a verification system.

Verification

The process of comparing two levels of system specification for proper correspondence (e.g., security policy model with top-level specification, top-level specification with source code, or source code with object code). This process may or may not be automated. [1]

Verification Committee

A standing committee responsible for the management of the verification efforts at the NCSC. The committee is chaired by the NCSC Deputy Director and includes the NCSC Chief Scientist, as well as representatives from both the NCSC's Office of Research and Development and Office of Computer Security Evaluations, Publications, and Support.

Verification System

An integrated set of tools and techniques for performing verification.

Well-Formed Formula

A sequence of symbols from a language that is constructed in accordance with the syntax for that language.

BIBLIOGRAPHY

[1] Department of Defense, Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, December 1985.

[2] Kemmerer, Richard A., Verification Assessment Study Final Report, University of California, March 1986.

[3] National Computer Security Center, A Guide to Understanding Configuration Management in Trusted Systems, NCSC-TG-006, March 1988.

[4] National Computer Security Center, Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria, NCSC-TG-005, July 1987.

[5] National Security Agency, Information Systems Security Products and Services Catalogue, Issued Quarterly, January 1989 and successors.