

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,



Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comments on Part1, Chapter 6, 6.4.1.3 Selection of general coding conventions

## From ASTWiki

The VVSG should require vendors to supply the coding standard to which their code complies, and the publications from which it is derived.

The VVSG should contain sections on minimum coding standards. For example:

## Contents

- 1 6.4.1.3-B Minimum Coding Standards
  - 1.1 6.4.1.3-B.1 Naming of procedures and data
    - 1.1.1 Example
    - 1.1.2 Example
  - 1.2 6.4.1.3-B.2 Parameterization of Constants
    - 1.2.1 Example
  - 1.3 6.4.1.3-B.3 Coding Comments
    - 1.3.1 Example

## 6.4.1.3-B Minimum Coding Standards

Vendor coding standards **SHALL** meet or exceed the minimum coding standards of the following sections:

### 6.4.1.3-B.1 Naming of procedures and data

All names of procedures and data **SHALL** be clearly and specifically descriptive of items they represent. Abbreviations **SHALL NOT** be allowed.

Names of procedures **SHALL** be imperative verb clauses except when returning a data value where the procedure **SHALL** be named to be descriptive of the data value returned.

Names of data items **SHALL** be noun clauses descriptive of the data items they represent.

#### Example

Review of the SAVIOC code (<http://www.savioc.com/SAVIOCsc.exe>) from SAVIOC Voting Systems (<http://www.savioc.com/>) reveals frequent if not total use of

- Abbreviations such as "Wlallowed", "ln", etc.
- Generic names such as "CharPtr", "templine", etc.
- Coded names such as "lc\_N", "MNprty", etc.
- Non descriptive procedure names such as "first\_pass", "party\_adds", etc.

#### Example

The code in the VVSG must conform to these requirements as well. See Part 1, Chapter 6.4.1.5-A.1 Legacy library units must be wrapped (<http://www.eac.gov/vvsg/part1/chapter06.php#6.4.1.5-A.1>) for an example of non-compliant variable names and data items.

### 6.4.1.3-B.2 Parameterization of Constants

Constants **SHALL** be named, defined and explained. Only names of constants **SHALL** be used in procedural code.

Constants **SHALL** be appropriately named as the data they represent.

#### Example

Review of the SAVIOC code (<http://www.savioc.com/SAVIOCsc.exe>) from SAVIOC Voting Systems (<http://www.savioc.com/>) reveals frequent use of unidentified constants:

- `char digit[3]; // Numeric character What is 3?`
- `allowed = 80-(locCselect); What is 80?`
- `window(1,25, 78,25); What are these numbers? What is the purpose and nature of this window?`

### 6.4.1.3-B.3 Coding Comments

Comments embedded in code **SHALL** be complete, grammatical English language sentences that explain the code to which they refer.

Comments embedded in code **SHALL NOT** contain abbreviations nor program identifiers (except parenthetically).

Comments embedded in code **SHALL** contain only ASCII printable and whitespace characters.

Comments **SHALL NOT** be used to disable the compilation of sections of code.

#### Example

See the non-compliant comments, constants, and variable names in the code in Part 3, Chapter 5.3.2 Critical values (<http://www.eac.gov/vvsg/part3/chapter05.php#5.3.2>) of the VVSG.

Retrieved from

"[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Part1%2C\\_Chapter\\_6%2C\\_6.4.1.3\\_Selection\\_o](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Part1%2C_Chapter_6%2C_6.4.1.3_Selection_o)

Category: Discussion

---

- This page was last modified 17:00, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

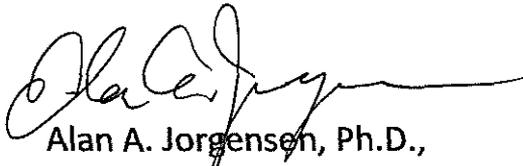
19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,



Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comment on Open Testing

## From ASTWiki

We recommend that the TGDC should open up the testing of eVoting systems to testers, free to work alone or in teams, who are willing to undergo a reasonable certification process, rather than restricting it to a small number of large companies and to only major test labs who are paid by the vendors, creating a clear conflict of interest between the public good and good publicity for the test lab's clients.

The problem of qualifying the voting system software was presented to the Workshop on Regulated Software Testing (WREST), held Nov. 16-17 2007 in Indianapolis, resulting in the identification of the following alternative approaches, each of which has its merits and its limits.

None of these approaches is risk free, none of them provides a complete solution.

They are presented here as recommendations to the TGDC.

1. Exhaustive preplanned testing of the software by an independent test lab.
2. Extensive exploratory testing of the software by an independent test lab.
3. Regulation of the coding practices, with process standards, adoption of good programming practices, rejection of bad programming practices, etc.
4. Public inspection of the code (open source)
5. Public protection via whistleblower protection
6. Accountability for process failures (prosecution or civil liability)
7. Accountability for product failures.

## Contents

- 1 Discussion
  - 1.1 Exhaustive testing using preplanned tests
  - 1.2 Independent test labs
- 2 Independent exploratory testing
- 3 Regulation of coding practices and development processes
- 4 Public inspection (Open Source)
- 5 Whistleblower protection
- 6 Accountability for process failures
- 7 Accountability for product failures

## Discussion

### Exhaustive testing using preplanned tests

This approach to software testing is extremely expensive and relatively ineffective. The cost of test documentation is enormous. The inertia created by the documentation with the maintenance burden associated with product change is enormous. The result is that system-level tests are written in a way to achieve traceability to written requirements, but without much emphasis on the power of the tests, where power is the ability to find a bug if it is there.

Even if the pool of tests was carefully designed to first maximize power, rerunning the same test means that the program is repaired in the ways that are exposed by those particular tests and other defects are overlooked. Given that any finite set of tests is a vanishingly small subset of the pool of possible distinct tests (two tests are distinct if the program could fail one while passing the other), running the same set tests over and over is a bad sampling strategy--but it is the norm for this approach to testing.

### Independent test labs

Test labs are certified by NIST but selected and paid by the equipment vendor. The test lab has an incentive to please the vendor and a requirement to meet the standards of NIST. The incentive to please the vendor involves

- (a) repeat business with this vendor and
- (b) recommendations (formally or grapevine) that lead to contracts with other vendors in this market space.

The optimal solution for the lab is likely to be to develop a testing approach (and suite of tests) that meet the specified-in-writing testing requirements (e.g. code coverage requirements, spec item coverage requirements) and not go further. It is relatively easy to write tests to meet coverage criteria that are not particularly powerful nor particularly creative.

We do not recommend elimination of testing by independent labs, but we recommend that this not be the primary mechanism for quality control.

## Independent exploratory testing

In the VVSG provisions for security testing, the required main style of testing is exploratory testing (aka "Open Ended Vulnerability Testing.") The term "exploratory testing" was coined 24 years ago by a recognized leader in the software testing field who been one of the leading advocates of this approach in mainstream system testing. The arguments that favor exploratory testing for security are the same for all other types of system-level tests: in the hands of a tester who is knowledgeable, motivated and skilled, this approach leads to a more varied set of harsh tests and a higher bug-find rate.

Note the three qualifiers: knowledgeable, motivated and skilled.

Without these qualifications, exploratory testing is simply undocumented ineffective testing.

It is not known how to regulate the quality of exploratory testing. The challenge of managing and coaching high-quality explorers is unsolved in the commercial testing community. It is subject to much discussion among practitioners (with almost no discussion among academics) with little more than a start on the solution.

The challenge becomes more difficult in a regulated environment with conflicts of interest. Most of the constructive practitioner discussion comes from people who have chosen to do most of their work in easier (commercial) environments.

## Regulation of coding practices and development processes

It is good to require that a development may not use a particular coding practice, even if that coding practice is available in the development language. But complying with that practice doesn't mean that a program is well written, it means that a bad program still conforms to that requirement.

It might not be a bad thing to say that process standards like CMMI must be followed, but the evidence that these standards actually substantially improve quality (or achieve sufficient-quality products) is mixed. One critical issue is motivation. A company that voluntarily adopts a process standard, any process standard, because its executives want to use this standard as their tool for managing the development costs and risks of their products is much more likely to achieve meaningful success with the standard than one whose executives adopt it because they are required by regulation to demonstrate conformance to it.

## Public inspection (Open Source)

The fact that the public CAN inspect open source code doesn't mean that the inspection will be done or that it will be done well. Even if several university labs are motivated to do this (for example) and get funding (e.g. NSF) to do this type of work, all of them go through learning curves. It might be that most of the testing actually performed is redundant and rather elementary.

Some of the larger open source products seem well tested, but to a significant degree, they are more well-used than well-tested. If they design their bug reporting systems well, then users will report failures as they happen. This is the main source of reports for the Firefox web browser, for example.

In contrast, the voting systems will not be well-used in the same way. The community of users of voting systems don't use them every day. They use them sporadically and only for mission-critical tasks. Most of the people who are likely to witness failures in the field are relatively inexperienced volunteers who are not trained in bug reporting and, under the circumstances of a live election, not welcome to engage in troubleshooting (e.g. trying to create a reproducible test case on a second machine) while the failure is

observable.

Another common objection to open source is the argument of security by obscurity; programs may have exploitable flaws, but they are not exploitable if no one knows about them. If we open the code to everyone, then Bad People might inspect the code for vulnerabilities and exploit them rather than reporting them. This argument has been widely challenged, but in an election system, we should assume that there are some people who have budget and motivation to find flaws that they could use to manipulate an election. We should also assume that these people will not reveal what flaws they discover, but exploit them. The public access protects against this only if the public testing is so thorough that the same bugs are likely to be found by people who will report them.

If public access to the source code is allowed, we must ensure that a lot of public testing is done, or we risk doing more harm than good. By making voting system software open source, testing and improvement is encouraged as there are people who will do these things just as they do today in the open source world. Consider GNU sourcing and Wikipedia.

There is already precedence for open sourcing in the government contracting world. Much of the software developed for NASA by companies is made available to the public domain. Contractors may not like this, but if they wish to business, with this customer set (those who buy voting machines), then it is a cost they must bear.

The voting system is a public "trust" and by making voting system software open source the public "trust" in electronic voting systems will be improved.

In addition, if public access to eVoting source code is allowed, it is essential that up-to-date voting equipment be available at reasonable cost to researchers (university or general public) who will test it.

## **Whistleblower protection**

VVSG offers no specific protection for whistleblowers, people who report serious problems with the voting product to government authorities or the public. Insiders know a lot about the weaknesses of a product. Enabling them to make disclosure creates a source of information that is probably much greater than an independent test lab can learn.

There are tremendous risks associated with being a whistleblower. Even if you have legal protection (a level of protection that is being steadily eroded in the USA), this type of disclosure is often career-ending. We cannot rely on it as a primary quality control process.

## **Accountability for process failures**

There is some accountability in place in the VVSG today. For example, with the adoption of VVSG, it is unlawful to load unapproved software into a voting system before an election (last minute patches, for example).

## **Accountability for product failures**

If voting system software/hardware fails in ways that result in miscounted votes or denial of service to some members of the community, they should be subject to accountability of some form.

Consider an analogy to medical devices. The manufacturer spends large amounts of money and effort conforming to regulations, but FDA approval is sufficient only to get to market. If the product fails in the field, the manufacturer is subject to products liability suits. The result is that many manufacturers adopt additional measures (such as extensive exploratory testing that is only lightly reported to FDA) to mitigate the risk of failure in the field, a very different risk from risk of non-certification by the regulator.

The primary benefit of products liability is that it provides additional incentive to make a product that actually works safely in the field. There are several risks, and products liability litigation, particularly for software failure, is politically unfashionable in the United States.

Retrieved from "[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comment\\_on\\_Open\\_Testing](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comment_on_Open_Testing)"

- This page was last modified 16:57, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

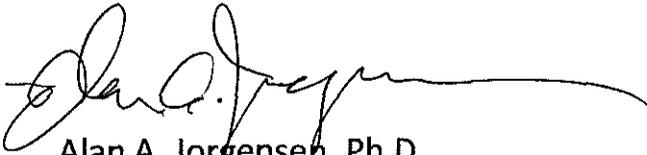
19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,

A handwritten signature in black ink, appearing to read "Alan A. Jorgensen", with a long horizontal flourish extending to the right.

Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comments on Part1, Chapter 3, 3.2.5-B Resetting of adjustable aspects at end of session

## From ASTWiki

VVSG 3.2.5-B Resetting of adjustable aspects at end of session (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.5-B>)

The intent of this requirement, as stated in the Discussion, is to ensure "that the voting station presents the same initial appearance to every voter." The true scope of the requirement though is much more fundamental and far-reaching: the system should present to every voter the same opportunity to vote for their choices. This includes resetting all alterable audio / video features at the end of every session, and also the invisible ones that computers specialize in:

- ensuring there is sufficient disk or storage space to record any valid combination of votes,
- ensuring that all traces of the previous vote have been removed from the display and from the memory sectors that will store the next votes
- ensuring all internal variables are reset to a neutral state
- etc...

Furthermore, for voter-verifiable paper audit trail (VVPAT) systems incorporating a printer and paper stock for voter-verifiable paper records (VVPRs), the system must perform a reliable self-test to ensure that the next voter will get a printed record, without paper jams, empty rolls, lack of toner, etc.

Software should self-test touchscreens between voting sessions to check for damage, such as areas of low voltage, and track trends in the differences between the coordinates of the icons or buttons to touch, and the actual spot touched on the screen. A consistent drift could trigger a call to adjust the screen.

Any tests that demonstrate that a system does not ALWAYS reset to a consistent pre-voting state will accurately predict unreliable performance in the field.

Retrieved from

["http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Part1%2C\\_Chapter\\_3%2C\\_3.2.5-B\\_Resetting\\_of\\_adjustable\\_aspects\\_at\\_end\\_of\\_session"](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Part1%2C_Chapter_3%2C_3.2.5-B_Resetting_of_adjustable_aspects_at_end_of_session)

Category: Discussion

- 
- This page was last modified 16:58, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,

A handwritten signature in black ink, appearing to read 'Alan A. Jorgensen', with a long horizontal line extending to the right.

Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaaj@gotsky.com](mailto:aaaj@gotsky.com)  
(321) 960-0109

# Comments on Part1, Chapter 6, 6.4.1.3 Selection of general coding conventions

## From ASTWiki

The VVSG should require vendors to supply the coding standard to which their code complies, and the publications from which it is derived.

The VVSG should contain sections on minimum coding standards. For example:

## Contents

- 1 6.4.1.3-B Minimum Coding Standards
  - 1.1 6.4.1.3-B.1 Naming of procedures and data
    - 1.1.1 Example
    - 1.1.2 Example
  - 1.2 6.4.1.3-B.2 Parameterization of Constants
    - 1.2.1 Example
  - 1.3 6.4.1.3-B.3 Coding Comments
    - 1.3.1 Example

## 6.4.1.3-B Minimum Coding Standards

Vendor coding standards **SHALL** meet or exceed the minimum coding standards of the following sections:

### 6.4.1.3-B.1 Naming of procedures and data

All names of procedures and data **SHALL** be clearly and specifically descriptive of items they represent. Abbreviations **SHALL NOT** be allowed.

Names of procedures **SHALL** be imperative verb clauses except when returning a data value where the procedure **SHALL** be named to be descriptive of the data value returned.

Names of data items **SHALL** be noun clauses descriptive of the data items they represent.

#### Example

Review of the SAVIOC code (<http://www.savioc.com/SAVIOCsc.exe>) from SAVIOC Voting Systems (<http://www.savioc.com/>) reveals frequent if not total use of

- Abbreviations such as "Wlallowed", "ln", etc.
- Generic names such as "CharPtr", "templne", etc.
- Coded names such as "lc\_N", "MNprty", etc.
- Non descriptive procedure names such as "first\_pass", "party\_adds", etc.

#### Example

The code in the VVSG must conform to these requirements as well. See Part 1, Chapter 6.4.1.5-A.1 Legacy library units must be wrapped (<http://www.eac.gov/vvsg/part1/chapter06.php#6.4.1.5-A.1>) for an example of non-compliant variable names and data items.

### 6.4.1.3-B.2 Parameterization of Constants

Constants **SHALL** be named, defined and explained. Only names of constants **SHALL** be used in procedural code.

Constants **SHALL** be appropriately named as the data they represent.

#### Example

Review of the SAVIOC code (<http://www.savioc.com/SAVIOCsc.exe>) from SAVIOC Voting Systems (<http://www.savioc.com/>) reveals frequent use of unidentified constants:

- `char digit[3]; // Numeric character What is 3?`
- `allowed = 80-(locCselect); What is 80?`
- `window(1,25, 78,25); What are these numbers? What is the purpose and nature of this window?`

### 6.4.1.3-B.3 Coding Comments

Comments embedded in code **SHALL** be complete, grammatical English language sentences that explain the code to which they refer.

Comments embedded in code **SHALL NOT** contain abbreviations nor program identifiers (except parenthetically).

Comments embedded in code **SHALL** contain only ASCII printable and whitespace characters.

Comments **SHALL NOT** be used to disable the compilation of sections of code.

#### Example

See the non-compliant comments, constants, and variable names in the code in Part 3, Chapter 5.3.2 Critical values (<http://www.eac.gov/vvsg/part3/chapter05.php#5.3.2>) of the VVSG.

Retrieved from

"[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Part1%2C\\_Chapter\\_6%2C\\_6.4.1.3\\_Selection\\_o](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Part1%2C_Chapter_6%2C_6.4.1.3_Selection_o)

Category: Discussion

---

- This page was last modified 17:00, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,



Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comments on Part1, Chapter 6, 6.4.1.4-B.1 Callable unit length limit

## From ASTWiki

VVSG 6.4.1.4-B.1 Callable unit length limit (<http://www.eac.gov/vvsg/part1/chapter06.php#6.4.1.4-B.1>)

The purpose of this requirement is to make source code more readable, a derived requirement from testability since the most efficient method of improving software quality is through source code review. Therefore this requirement must read "SHALL" instead of "SHOULD".

While constraining the number of lines in a callable unit is an admirable goal, the real issue is cohesion ([http://en.wikipedia.org/wiki/Cohesion\\_%28computer\\_science%29](http://en.wikipedia.org/wiki/Cohesion_%28computer_science%29)).

For the reasons cited in this reference, voting system software callable units must have high cohesion.

Methods for achieving high cohesion for a callable unit include:

- Minimize the number of data structures affected by that callable unit.
- Minimize the number of calls to other callable units.

We recommend that this VVSG requirement be more stringent by not allowing any callable units to exceed 50 lines in length (excluding comments, blank lines, and initializers for read-only lookup tables) by using the above methods for maximizing cohesion.

Retrieved from

"[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Part1%2C\\_Chapter\\_6%2C\\_6.4.1.4-B.1\\_Callabl](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Part1%2C_Chapter_6%2C_6.4.1.4-B.1_Callabl)

Category: Discussion

- 
- This page was last modified 17:01, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

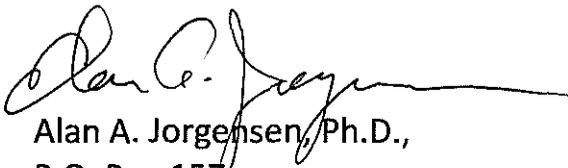
19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,



Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comments on Part1, Chapter 6, 6.4.1.8-B Mandatory internal error checking

From ASTWiki

VVSG 6.4.1.8-B Mandatory internal error checking (<http://www.eac.gov/vvsg/part1/chapter06.php#6.4.1.8-B>)

## 1

We recommend adding an additional numbered item:

### 7. Numeric Underflows

Numeric underflows are a well-understood source of possible system failures, and there should be a requirement to test for them as well.

For example, in the SAVIOC voting system code, during the releasing of previously allocated memory for ballots, in order to allocate new memory for other purposes, an index of the number of allocated ballots is decremented without checking. When no more ballots are available for deallocation, the index may go negative, an underflow, causing serious failure of the voting system.

## 2

For each bulleted item of Section 6.4.1.8-B there is a corresponding requirement, except for bullet number 6. There must be a requirement that any known vulnerabilities are mitigated and that testing for those vulnerabilities must take place.

Suggested content of sub-sections B-6 and B-7:

### 6.4.1.8-B.6 Known programming language specific vulnerabilities

(Include in this section examples of specific programming language vulnerabilities, for instance, the C and C++ if-then-else ambiguity.)

### 6.4.1.8-B.7 Numeric Underflows

All arithmetic operations that could potentially underflow the relevant data type *SHALL* be checked for underflow.

Retrieved from

"[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Part1%2C\\_Chapter\\_6%2C\\_6.4.1.8-B\\_Mandator](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Part1%2C_Chapter_6%2C_6.4.1.8-B_Mandator)

Category: Discussion

- 
- This page was last modified 17:02, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,



Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comments on Part2, Chapter 2, 2.1-A.5 Problem log

From ASTWiki

VVSG 2.1-A.5 Problem log (<http://www.eac.gov/vvsg/part2/chapter02.php#2.1-A.5>)

1

We are unclear on the audience and purpose of this document. The requirement must spell out both in more detail, since the document written by a development team for testers in a remote test lab will be entirely different from a document written by a development team for themselves, or for a VVSG compliance auditor.

2

This requirement requires "all difficulties" and "any remedial actions" be entered in a log, which can mean an essentially infinite workload given the nature of software development projects. We recommend that this requirement clearly establish for the documenter the scope of information to be included, such as:

"The log SHALL be sufficient to expose the history of the system to the development team, so they can avoid, and test for, problems arising from remnants and artifacts of prior design decisions."

or

"The log SHALL be sufficient to enable an independent development process review AND to guide an independent code review seeking evidence of refactoring errors."

This offers reasonable guidance as to the scope of information, and also guides the intended audience in their use of the document so they can know how much information to expect.

3

The DISCUSSION section attempts to address the above issue by stating that only "difficulties" need to be logged, and defines "difficulties" as "events that force a change in plan or design". This definition is highly problematic because it places pressure on the vendor to reduce the level of detail in their plans and designs so that they can accommodate the natural flow of project work without change, since change burdens the team with excessive documentation. We recommend removing this definition entirely and adding a clear statement of purpose, the intended audience, and the appropriate "stopping heuristic" outlined above.

4

This requirement uses the phrase "design and development". The term "development" is ambiguous in many software "development" environments and often includes design. A better phrase would be "design and implementation" to ensure a clear distinction between design effort and implementation (coding).

Retrieved from

"[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Part2%2C\\_Chapter\\_2%2C\\_2.1-A.5\\_Problem\\_lc](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Part2%2C_Chapter_2%2C_2.1-A.5_Problem_lc)

Category: Discussion

---

- This page was last modified 17:04, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,



Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comments on Introduction, Chapter 2, 2.7 Treatment of COTS in Voting System Testing

From ASTWiki

VVSG 2.7 Treatment of COTS in Voting System Testing (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.7>)

Voting systems are complex enough, with their limited scope and well-defined user base, that many problems are not being found today during vendor or independent-lab testing. The VVSG as written does not make similar failures less likely in the future, since it mandates extremely expensive, documentation-intensive (as opposed to test-intensive) test processes for both the vendor and the independent test lab. The closed-source nature of the voting system code and the difficulty a private person has in acquiring voting systems to test makes it impossible for industry outsiders such as the Association for Software Testing or any other concerned citizen to lend a hand in the testing effort. So we recommend that the VVSG mandate all voting machines run open-source code for all aspects of its functioning.

The treatment of COTS in the VVSG encourages vendors to use "safe", "commonly available" utilities for various functions, without restriction on the licensing of the COTS code. This leaves gaping security, reviewability, and testability holes in the voting software. The VVSG baselessly claims that a COTS utility, if procured and installed successfully by the tester, must therefore pose no threat to the security of the system. It is impossible to assess the fitness of the utility for its purpose if the code is closed. It is poor practice to include in a system functioning code which is intended never to be used, because it may be used unintentionally, with bad results. Most COTS products are designed and built for a wide base of users, for a wide list of use cases, and so are not built to do one thing in one way and only that way. It is in all the other unused capabilities of operating systems, browsers, comm stacks, etc. that bugs and security holes hide. So we advocate mandating open-source COTS utilities wherever possible.

Where that is not possible, the very least that must be done to evaluate the suitability and security of a COTS product used in voting software are:

- The VVSG must mandate that binaries of COTS products be made available to testers.
- There must be no restrictions on reverse-engineering the COTS product and replacing it with a test stub.
- Voting software must wrap the COTS code, isolating it as much as possible from interacting with the OS except through the aperture of reviewable voting code.
- A list of viable alternative COTS products must be provided as well as a method for substitution.
- The complete list of available functions and interfaces for the utility must be provided, not just those that are used by the voting system.

Retrieved from

"[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Introduction%2C\\_Chapter\\_2%2C\\_2.7\\_Treatment](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Introduction%2C_Chapter_2%2C_2.7_Treatment)

Category: Discussion

- 
- This page was last modified 17:05, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,

A handwritten signature in black ink, appearing to read "Alan A. Jorgensen", with a long horizontal flourish extending to the right.

Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comments on Part3, Chapter 2, 2.2 Scope of Assessment

From ASTWiki

VVSG 2.2 Scope of Assessment (<http://www.eac.gov/vvsg/part3/chapter02.php#2.2>)

This long section makes the statement;

"Not all systems are required to complete every category of testing. Consistent with Requirement Part 2: 5.1-D, the test lab may find that proven performance of COTS hardware, software and communications components in commercial applications other than elections obviates the need for certain specific evaluations."

This statement assumes "proven performance" establishes applicability to the eVoting application. This is not the case.

This section must be changed to require that all COTS software be tested with the same rigor as the other components in the system.

## Published Example

The document "Software Pest Control Suite (SPCS) User Manual", Alan A. Jorgensen and Kim McCarter, 2001, submitted as part of the final report for U. S. Army SBIR contract #DAAD17-99-C-0055, "Automated Testing of Reusable ADA Software Components" reveals a flaw in the Microsoft "libc" components "fprintf" and "fscanf", the basic file formatted write and read functions.

Under certain conditions, these functions fail to return error indicators when the function calls are used improperly, specifically, when a file is open for reading, the "fprintf" command will cause the file position variable to be changed causing data to be skipped during read of an input file, and the "fscanf" command likewise advances file position causing trash to be written into the output file.

These defects are very difficult to detect and diagnose in the context of system testing and were actually found during state model based component testing of the Microsoft "libc" "stdio" functions. (The IBM libc passed the tests, however, the open source GNU libc failed the tests in a different way.)

## Code Example

The following program fails when compiled and linked with,

```
-----  
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8168 for 80x86  
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.  
TestStdio.c  
Microsoft (R) Incremental Linker Version 6.00.8168  
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.  
-----
```

though it appears to the code reviewer that it should work without any trouble:

```

/*      This program was reformatted by prettyc Rev. 2.2      */
#include <stdio.h>
main()
{
    FILE * TestFile;
    int ErrorCode;
    char InputData[1000];
    printf("\nTesting fscanf when file open for write only.\n\n");

    /*
    ** Open file for write
    */
    TestFile = fopen("TestFile", "w");
    if (TestFile == NULL)
        printf("Failed to open TestFile for writing.\n");

    /*
    ** Write Data to file
    */
    ErrorCode = fprintf(TestFile, "This is test data.\n");
    if (ErrorCode == - 1)
        printf("Write to TestFile returned error code.\n");

    /*
    ** Illegally read from file
    */
    ErrorCode = fscanf(TestFile, InputData);
    if (ErrorCode != - 1)
        printf(
            "\nfscanf failed to return error code for file open for write.\n")
        ;

    /*
    ** Write Data to file
    */
    ErrorCode = fprintf(TestFile, "This is test data.\n");
    if (ErrorCode == - 1)
        printf("Write to TestFile returned error code.\n");

    /*
    ** Close File
    */
    ErrorCode = fclose(TestFile);
    if (ErrorCode == - 1)
        printf("Closing file resulted in error.\n");

    /*
    ** Try Closing File again
    */
    ErrorCode = fclose(TestFile);
    if (ErrorCode != - 1)
        printf("Closing closed file did not return error.\n");
    printf("\nThe contents of the test file:\n\n");

    /*
    ** Display the contents of the Test File
    */
    system("type TestFile\n");
}

```

Retrieved from

"[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Part3%2C\\_Chapter\\_2%2C\\_2.2\\_Scope\\_of\\_Asses](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Part3%2C_Chapter_2%2C_2.2_Scope_of_Asses)

Category: Discussion

- This page was last modified 17:06, 23 March 2008.

U.S. Election Assistance Commission  
1225 New York Avenue NW, Suite 1100,  
Washington DC, 20005

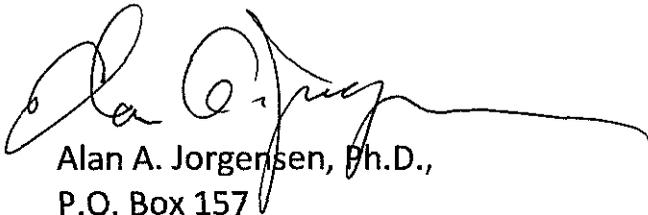
19 April 2008

To Whom it May Concern,

The attached comment on the 2007 version of the Voluntary Voting Systems Guidelines is being submitted by hard copy Postal Service mail because the on-line submission form does not permit formatting of submissions.

We strongly recommend that this shortcoming be rectified.

Submitted for the  
Association for Software Testing  
Special Interest Group on Electronic Voting  
by,

A handwritten signature in black ink, appearing to read 'Alan A. Jorgensen', with a long horizontal flourish extending to the right.

Alan A. Jorgensen, Ph.D.,  
P.O. Box 157  
Seligman, Arizona 86337  
[aaj@gotsky.com](mailto:aaj@gotsky.com)  
(321) 960-0109

# Comments on Appendix A: Definitions of Words with Special Meanings

## From ASTWiki

VVSG Definitions of Words with Special Meanings (<http://www.eac.gov/vvsg/g>)

The following terms have special meaning within the context of the VVSG and need formal definition.

### Acceptance Testing, Acceptance Tests

This term appears in:

- Part 1, Chapter 8, 8.1.3 8.1.3 Translation of diagrams (<http://www.eac.gov/vvsg/part1/chapter08.php#8.1.3>)
- Part 2, Chapter 3, 3.1.2 3.1.2 Other uses for documentation (<http://www.eac.gov/vvsg/part2/chapter03.php#3.1.2>)
- Part 3, Chapter 1, 1.1.4.1 1.1.4.1 End-to-End testing (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.4.1>)

### Accredited

We recommend that the guide not limit testing to only "accredited" test labs (See our recommendations on the open source issue). The guidelines must define the base minimal acceptable levels of testing, and then include wording that encourages testing above that minimal level. The minimal level as currently defined includes "an independent accredited" test lab. However, a better definition of accredited is needed:

- Who does the accreditation?
- How is accreditation accomplished?
- When is accreditation performed?
- How long is accreditation valid?
- Etc.

This term appears in:

- Introduction, Chapter 3, 3.5 3.5 Relationship of HAVA and the VVSG (<http://www.eac.gov/vvsg/introduction/chapter03.php#3.5>)
- Part 1, Chapter 1, 1.1.11 1.1.11 Supplemental Guidance (<http://www.eac.gov/vvsg/part1/chapter01.php#1.1.11>)
- Part 2, Chapter 3, 3.1 3.1 Scope (<http://www.eac.gov/vvsg/part2/chapter03.php#3.1>)
- Part 2, Chapter 3, 3.1.3-A 3.1.3-A TDP, identify proprietary data (<http://www.eac.gov/vvsg/part2/chapter03.php#3.1.3-A>)
- Part 2, Chapter 3, 3.7-A 3.7-A TDP, system change notes (<http://www.eac.gov/vvsg/part2/chapter03.php#3.7-A>)
- Part 2, Chapter 5, 5.1-D 5.1-D Test plan, previous work (<http://www.eac.gov/vvsg/part2/chapter05.php#5.1-D>)
- Part 3, Chapter 2, 2.1 2.1 Overview (<http://www.eac.gov/vvsg/part3/chapter02.php#2.1>)
- Part 3, Chapter 2, 2.2 2.2 Scope of Assessment (<http://www.eac.gov/vvsg/part3/chapter02.php#2.2>)
- Part 3, Chapter 2, 2.4.1 2.4.1 Initiation of testing (<http://www.eac.gov/vvsg/part3/chapter02.php#2.4.1>)
- Part 3, Chapter 2, 2.4.2 2.4.2 Pre-test preparation (<http://www.eac.gov/vvsg/part3/chapter02.php#2.4.2>)
- Part 3, Chapter 2, 2.4.2.1-A 2.4.2.1-A Submit Technical Data Package (<http://www.eac.gov/vvsg/part3/chapter02.php#2.4.2.1-A>)
- Part 3, Chapter 2, 2.5.1-A 2.5.1-A Prepare test plan (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.1-A>)
- Part 3, Chapter 2, 2.5.5-A 2.5.5-A Conduct all tests (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-A>)
- Part 3, Chapter 2, 2.5.5-F 2.5.5-F Pauses in test campaign (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-F>)
- Part 3, Chapter 2, 2.6.3-A 2.6.3-A Prepare test report (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.3-A>)
- Part 3, Chapter 2, 2.6.3-B 2.6.3-B Consolidated test report (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.3-B>)
- Part 3, Chapter 3, 3.1 3.1 Inspection (<http://www.eac.gov/vvsg/part3/chapter03.php#3.1>)
- Part 3, Chapter 4, 4.1 4.1 Initial Review of Documentation (<http://www.eac.gov/vvsg/part3/chapter04.php#4.1>)
- Part 3, Chapter 5, 5.2 5.2 Functional Testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2>)

## Accredited expert

This term appears in:

- Definitions of Words with Special Meanings (<http://www.eac.gov/vvsg/g/definitions-of-words-with-special-meanings/>)

## Accredited laboratory

This term appears in:

- Introduction, Chapter 3, 3.5 3.5 Relationship of HAVA and the VVSG (<http://www.eac.gov/vvsg/introduction/chapter03.php#3.5>)

## Accredited specialist

This term appears in:

- Part 3, Chapter 3, 3.1 3.1 Inspection (<http://www.eac.gov/vvsg/part3/chapter03.php#3.1>)

## Accredited test lab

This term appears in:

- Part 1, Chapter 1, 1.1.11 1.1.11 Supplemental Guidance (<http://www.eac.gov/vvsg/part1/chapter01.php#1.1.11>)
- Part 2, Chapter 3, 3.1 3.1 Scope (<http://www.eac.gov/vvsg/part2/chapter03.php#3.1>)
- Part 2, Chapter 3, 3.1.3-A 3.1.3-A TDP, identify proprietary data (<http://www.eac.gov/vvsg/part2/chapter03.php#3.1.3-A>)
- Part 2, Chapter 3, 3.7-A 3.7-A TDP, system change notes (<http://www.eac.gov/vvsg/part2/chapter03.php#3.7-A>)
- Part 2, Chapter 5, 5.1-D 5.1-D Test plan, previous work (<http://www.eac.gov/vvsg/part2/chapter05.php#5.1-D>)
- Part 3, Chapter 2, 2.1 2.1 Overview (<http://www.eac.gov/vvsg/part3/chapter02.php#2.1>)
- Part 3, Chapter 2, 2.2 2.2 Scope of Assessment (<http://www.eac.gov/vvsg/part3/chapter02.php#2.2>)
- Part 3, Chapter 2, 2.4.1 2.4.1 Initiation of testing (<http://www.eac.gov/vvsg/part3/chapter02.php#2.4.1>)
- Part 3, Chapter 2, 2.4.2 2.4.2 Pre-test preparation (<http://www.eac.gov/vvsg/part3/chapter02.php#2.4.2>)
- Part 3, Chapter 2, 2.4.2.1-A 2.4.2.1-A Submit Technical Data Package (<http://www.eac.gov/vvsg/part3/chapter02.php#2.4.2.1-A>)
- Part 3, Chapter 2, 2.5.1-A 2.5.1-A Prepare test plan (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.1-A>)
- Part 3, Chapter 2, 2.5.5-A 2.5.5-A Conduct all tests (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-A>)
- Part 3, Chapter 2, 2.5.5-F 2.5.5-F Pauses in test campaign (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-F>)
- Part 3, Chapter 2, 2.6.3-A 2.6.3-A Prepare test report (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.3-A>)
- Part 3, Chapter 2, 2.6.3-B 2.6.3-B Consolidated test report (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.3-B>)
- Part 3, Chapter 4, 4.1 4.1 Initial Review of Documentation (<http://www.eac.gov/vvsg/part3/chapter04.php#4.1>)
- Part 3, Chapter 5, 5.2 5.2 Functional Testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2>)

## Accuracy Testing

This term appears in:

- Part 1, Chapter 7, 7.3.1-J 7.3.1-J L&A testing, no side-effects (<http://www.eac.gov/vvsg/part1/chapter07.php#7.3.1-J>)
- Part 1, Chapter 7, 7.3.1 7.3.1 Logic and accuracy testing (<http://www.eac.gov/vvsg/part1/chapter07.php#7.3.1>)
- Part 1, Chapter 7, 7.8.2-C 7.8.2-C Status reports (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.2-C>)
- Part 2, Chapter 4, 4.3.5-A 4.3.5-A User documentation, model setup inspection process (<http://www.eac.gov/vvsg/part2/chapter04.php#4.3.5-A>)
- Part 2, Chapter 4, 4.4.3-A 4.4.3-A Operations manual, readiness testing (<http://www.eac.gov/vvsg/part2/chapter04.php#4.4.3-A>)
- Part 3, Chapter 4, 4.5.1-D 4.5.1-D Efficacy of built-in self-tests (<http://www.eac.gov/vvsg/part3/chapter04.php#4.5.1-D>)

## **Certified Testing Laboratory**

What is the distinction between "Accredited Testing Laboratory" and "Certified Testing Laboratory?"

This term appears in:

- Introduction, Chapter 2, 2.1.1 2.1.1 VVSG Standards Architecture (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.1.1>)

## **Conformance Testing**

This term appears in:

- Introduction, Chapter 3, 3.5 3.5 Relationship of HAVA and the VVSG (<http://www.eac.gov/vvsg/introduction/chapter03.php#3.5>)
- Part 1, Chapter 3, 3.2.1.2-A 3.2.1.2-A Usability testing by manufacturer for general population (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.1.2-A>)
- Part 3, Chapter 3, 3.5 3.5 Interoperability Testing (<http://www.eac.gov/vvsg/part3/chapter03.php#3.5>)

## **End-End**

The term "end-to-end" is defined in the glossary, however, in some places in the VVSG it is "end-end".

This term appears in:

- Introduction, Chapter 2, 2.2 Introduction to New and Expanded Material (<http://www.eac.gov/vvsg/introduction/chapter02.php#2>)

## **Functional testing**

This term is defined internally in the VVSG in Part3, Chapter 5, 5.2 Functional Testing as follows:

"Functional testing is the determination through operational testing of whether the behavior of a system or device in specific scenarios conforms to requirements."

This definition should also appear in the glossary.

This term appears in at least 240 sections of the VVSG.

## **Glass-box or Glass-box Testing**

Is this term equivalent to "White-box" testing?

This term appears in:

- Part 3, Chapter 1, 1.1.3.2 1.1.3.2 Logic verification (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.3.2>)
- Part 3, Chapter 5, 5.2.2 5.2.2 Structural coverage (white-box testing) (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.2>)

## **Independent Testing**

This term appears in:

- Part 2, Chapter 3, 3.5.1-B 3.5.1-B TDP, high level security (<http://www.eac.gov/vvsg/part2/chapter03.php#3.5.1-B>)
- Part 3, Chapter 5, 5.5 Test Methods (<http://www.eac.gov/vvsg/part3/chapter05.php#5>)

## **Interface Testing**

This term appears in:

- Part 3, Chapter 5, 5.2.2-B 5.2.2-B Interface testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.2-B>)
- Part 3, Chapter 5, 5.2.2 5.2.2 Structural coverage (white-box testing) (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.2>)

## Interoperability Testing

This term is defined internally in the VVSG in Part3, Chapter 3, 3.5 Interoperability Testing as:

"Interoperability testing is the determination through operational testing of whether existing products are able to cooperate meaningfully for some purpose. It consists of bringing together existing products, configuring them to work together, and performing a functional test to determine whether the operation succeeds."

A definition for this term should also appear in the glossary.

This term appears in:

- Part 1, Chapter 2, 2.7.2 2.7.2 Innovation class submissions (<http://www.eac.gov/vvsg/part1/chapter02.php#2.7.2>)
- Part 1, Chapter 6, 6.6-A 6.6-A Integrability of systems and devices (<http://www.eac.gov/vvsg/part1/chapter06.php#6.6-A>)
- Part 1, Chapter 6, 6.6-B 6.6-B Data export and exchange format (<http://www.eac.gov/vvsg/part1/chapter06.php#6.6-B>)
- Part 3, Chapter 3, 3.5 3.5 Interoperability Testing (<http://www.eac.gov/vvsg/part3/chapter03.php#3.5>)

## Logic verification

This term appears in:

- Introduction, Chapter 2, 2.10 2.10 Expanded Core Requirements Coverage (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.10>)
- Part 1, Chapter 1, 1.1.9 1.1.9 Reference models (<http://www.eac.gov/vvsg/part1/chapter01.php#1.1.9>)
- Part 1, Chapter 6, 6.1-B 6.1-B Verifiably correct vote recording and tabulation (<http://www.eac.gov/vvsg/part1/chapter06.php#6.1-B>)
- Part 1, Chapter 6, 6.3.2-A 6.3.2-A Satisfy integrity constraints (<http://www.eac.gov/vvsg/part1/chapter06.php#6.3.2-A>)
- Part 1, Chapter 6, 6.4.1.8-B 6.4.1.8-B Mandatory internal error checking (<http://www.eac.gov/vvsg/part1/chapter06.php#6.4.1.8-B>)
- Part 1, Chapter 7, 7.7.2-A 7.7.2-A Tabulator, voting variations (<http://www.eac.gov/vvsg/part1/chapter07.php#7.7.2-A>)
- Part 1, Chapter 7, 7.8.3.1-C 7.8.3.1-C Account for all cast ballots and all valid votes (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.1-C>)
- Part 1, Chapter 7, 7.8.3.1-D 7.8.3.1-D Vote data reports, discrepancies can't happen (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.1-D>)
- Part 1, Chapter 7, 7.8.3.2-B 7.8.3.2-B Report read ballots (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.2-B>)
- Part 1, Chapter 7, 7.8.3.2-C 7.8.3.2-C Report counted ballots (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.2-C>)
- Part 1, Chapter 7, 7.8.3.2-D 7.8.3.2-D Report counted ballots by contest (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.2-D>)
- Part 1, Chapter 7, 7.8.3.3-A 7.8.3.3-A Report votes for each contest choice (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.3-A>)
- Part 1, Chapter 7, 7.8.3.3-B 7.8.3.3-B Report overvotes for each contest (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.3-B>)
- Part 1, Chapter 7, 7.8.3.3-C 7.8.3.3-C Report undervotes for each contest (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.3-C>)
- Part 1, Chapter 7, 7.8.3.3-E 7.8.3.3-E Include in-person votes (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.3-E>)
- Part 1, Chapter 7, 7.8.3.3-F 7.8.3.3-F Include absentee votes (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.3-F>)
- Part 1, Chapter 7, 7.8.3.3-G 7.8.3.3-G Include write-in votes (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.3-G>)
- Part 1, Chapter 7, 7.8.3.3-H 7.8.3.3-H Include accepted provisional-challenged votes (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.3-H>)
- Part 1, Chapter 7, 7.8.3.3-I 7.8.3.3-I Include accepted reviewed votes (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.3-I>)
- Part 1, Chapter 8, 8.3.1 8.3.1 Domain of discourse (<http://www.eac.gov/vvsg/part1/chapter08.php#8.3.1>)

- Part 2, Chapter 1, 1.1.2 1.1.2 Changes in TDP content (<http://www.eac.gov/vvsg/part2/chapter01.php#1.1.2>)
- Part 2, Chapter 3, 3.4.7.2-F 3.4.7.2-F TDP, inductive assertions (<http://www.eac.gov/vvsg/part2/chapter03.php#3.4.7.2-F>)
- Part 3, Chapter 1, 1.1.2 1.1.2 Applicability to COTS and borderline COTS products (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.2>)
- Part 3, Chapter 1, 1.1.3.2 1.1.3.2 Logic verification (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.3.2>)
- Part 3, Chapter 3, 3.1 3.1 Inspection (<http://www.eac.gov/vvsg/part3/chapter03.php#3.1>)
- Part 3, Chapter 4, 4.6 4.6 Logic Verification (<http://www.eac.gov/vvsg/part3/chapter04.php#4.6>)
- Part 3, Chapter 5, 5.2.3-B.1 5.2.3-B.1 Practical limit on capacity operational tests (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.3-B.1>)
- Part 3, Chapter 5, 5.3.1 5.3.1 General method (<http://www.eac.gov/vvsg/part3/chapter05.php#5.3.1>)

### **Manufacturer Testing**

This term appears in:

- Part 1, Chapter 3, 3.2.1.2 3.2.1.2 Manufacturer testing (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.1.2>)
- Part 3, Chapter 5, 5 5 Test Methods (<http://www.eac.gov/vvsg/part3/chapter05.php#5>)

### **National Certification Authority**

This term appears in:

- Introduction, Chapter 1, 1.3 1.3 Audience (<http://www.eac.gov/vvsg/introduction/chapter01.php#1.3>)
- Part 3, Chapter 2, 2.6.2.1-A.5 2.6.2.1-A.5 Master copy retention (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.2.1-A.5>)
- Part 3, Chapter 2, 2.6.2.2-B 2.6.2.2-B Repository digital signature verification (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.2.2-B>)
- Part 3, Chapter 2, 2.6.2.2-C 2.6.2.2-C Repository software distribution package (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.2.2-C>)
- Part 3, Chapter 2, 2.6.2.2-D 2.6.2.2-D Notary repositories software integrity information software distribution package (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.2.2-D>)
- Part 3, Chapter 2, 2.6.2.2 2.6.2.2 Repository software distribution requirements (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.2.2>)
- Part 3, Chapter 2, 2.6.2.3-C 2.6.2.3-C Software distribution packages for manufacturers, National Software Reference Library (NSRL), and designated national repository (<http://www.eac.gov/vvsg/part3/chapter02.php#2.6.2.3-C>)

### **National Certification Process**

This term appears in:

- Introduction, Chapter 1, 1.3 1.3 Audience (<http://www.eac.gov/vvsg/introduction/chapter01.php#1.3>)

### **National Certification Testing**

This term appears in:

- Introduction, Chapter 2, 2.1.1 2.1.1 VVSG Standards Architecture (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.1.1>)
- Part 1, Chapter 6, 6.3.4.3-A 6.3.4.3-A Power port disturbances (<http://www.eac.gov/vvsg/part1/chapter06.php#6.3.4.3-A>)

### **Open-Ended Testing**

This term is used with a modifier, "vulnerability" as in "Open Ended Vulnerability Testing".

Our recommendation is that this term be replaced in the VVSG with the commonly accepted term: "Exploratory Testing".

Sometimes in the VVSG "Open-Ended" is hyphenated, sometimes it is not.

### **Penetration Testing**

This term appears in:

- Part 1, Chapter 7, 7.5.1 7.5.1 Issuance of voting credentials and ballot activation (<http://www.eac.gov/vvsg/part1/chapter07.php#7.5.1>)
- Part 3, Chapter 1, 1.1.4.3 1.1.4.3 Open-ended vulnerability testing (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.4.3>)
- Part 3, Chapter 3, 3.4 3.4 Vulnerability Testing (<http://www.eac.gov/vvsg/part3/chapter03.php#3.4>)
- Part 3, Chapter 5, 5.4.2-C 5.4.2-C OEVT team composition – security experts (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.2-C>)

### **Performance-based**

This term appears in:

- Introduction, Chapter 2, 2.2 2.2 Usability Performance Requirements (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.2>)
- Introduction, Chapter 3, 3.4 3.4 HAVA and VVSG 2005 (<http://www.eac.gov/vvsg/introduction/chapter03.php#3.4>)
- Part 1, Chapter 3, 3.2.1 3.2.1 Performance Requirements (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.1>)

### **Performance Testing**

This term appears in:

- Introduction, Chapter 2, 2.2 2.2 Usability Performance Requirements (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.2>)
- Part 2, Chapter 6, 6.1 6.1 Test report contents (<http://www.eac.gov/vvsg/part2/chapter06.php#6.1>)
- Part 3, Chapter 2, 2.2 2.2 Scope of Assessment (<http://www.eac.gov/vvsg/part3/chapter02.php#2.2>)
- Part 3, Chapter 3, 3.3 3.3 Performance Testing (Benchmarking) (<http://www.eac.gov/vvsg/part3/chapter03.php#3.3>)

### **Plain Language**

This term appears in:

- Introduction, Chapter 2, 2.3 2.3 Expanded Usability and Accessibility Coverage (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.3>)
- Part 1, Chapter 3, 3.2.4-C 3.2.4-C Plain Language (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.4-C>)
- Part 1, Chapter 3, 3.3.7-A 3.3.7-A General support for cognitive disabilities (<http://www.eac.gov/vvsg/part1/chapter03.php#3.3.7-A>)
- Part 1, Chapter 6, 6.3.1.3 6.3.1.3 Manageable failures per election (<http://www.eac.gov/vvsg/part1/chapter06.php#6.3.1.3>)
- Part 1, Chapter 7, 7.7.5-D 7.7.5-D MCOS, accurately detect imperfect marks (<http://www.eac.gov/vvsg/part1/chapter07.php#7.7.5-D>)

### **Qualification Testing**

This term appears in:

- Part 3, Chapter 2, 2.1 2.1 Overview (<http://www.eac.gov/vvsg/part3/chapter02.php#2.1>)

## Readiness Testing

This term appears in:

- Part 2, Chapter 4, 4.4.3-A 4.4.3-A Operations manual, readiness testing (<http://www.eac.gov/vvsg/part2/chapter04.php#4.4.3-A>)
- Part 2, Chapter 4, 4.4.7-A 4.4.7-A Operations manual, operations support (<http://www.eac.gov/vvsg/part2/chapter04.php#4.4.7-A>)

## Regression Testing

This term appears in:

- Part 3, Chapter 2, 2.3 2.3 Testing Sequence (<http://www.eac.gov/vvsg/part3/chapter02.php#2.3>)
- Part 3, Chapter 2, 2.5.5-D 2.5.5-D Software defects are not field-serviceable (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-D>)
- Part 3, Chapter 5, 5.2.2 5.2.2 Structural coverage (white-box testing) (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.2>)
- Part 3, Chapter 5, 5.2.3 5.2.3 Functional coverage (black-box testing) (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.3>)

## Security Testing

This term appears in:

- Introduction, Chapter 2, 2.5 2.5 Open-Ended Vulnerability Testing (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.5>)
- Part 2, Chapter 3, 3.5.1-B 3.5.1-B TDP, high level security (<http://www.eac.gov/vvsg/part2/chapter03.php#3.5.1-B>)
- Part 3, Chapter 1, 1.1.4.3 1.1.4.3 Open-ended vulnerability testing (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.4.3>)
- Part 3, Chapter 5, 5.4 5.4 Open-Ended Vulnerability Testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4>)

## Software Performance

This term appears in:

- Part 1, Chapter 6, 6.3 6.3 Hardware and Software Performance, General Requirements (<http://www.eac.gov/vvsg/part1/chapter06.php#6.3>)

## Test Campaign

This term appears in:

- Introduction, Chapter 2, 2.9 2.9 Reliability (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.9>)
- Part 2, Chapter 5, 5.1-D 5.1-D Test plan, previous work (<http://www.eac.gov/vvsg/part2/chapter05.php#5.1-D>)
- Part 2, Chapter 6, 6.1-I 6.1-I Test report, anomalies (<http://www.eac.gov/vvsg/part2/chapter06.php#6.1-I>)
- Part 3, Chapter 1, 1.1.4.2 1.1.4.2 Reliability, accuracy, and probability of misfeed (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.4.2>)
- Part 3, Chapter 2, 2.3 2.3 Testing Sequence (<http://www.eac.gov/vvsg/part3/chapter02.php#2.3>)
- Part 3, Chapter 2, 2.5.5-C 2.5.5-C Critical software defects are unacceptable (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-C>)

- Part 3, Chapter 2, 2.5.5-D 2.5.5-D Software defects are not field-serviceable (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-D>)
- Part 3, Chapter 2, 2.5.5-E 2.5.5-E Hardware failures are field-serviceable (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-E>)
- Part 3, Chapter 2, 2.5.5-F 2.5.5-F Pauses in test campaign (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-F>)
- Part 3, Chapter 2, 2.5.5-G 2.5.5-G Resumption after deficiency (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.5-G>)
- Part 3, Chapter 5, 5.2.3-A 5.2.3-A Functional testing, VVSG requirements (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.3-A>)
- Part 3, Chapter 5, 5.3.1 5.3.1 General method (<http://www.eac.gov/vvsg/part3/chapter05.php#5.3.1>)

## Test Fixture

This term appears in:

- Part 3, Chapter 1, 1.1.4.1 1.1.4.1 End-to-End testing (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.4.1>)
- Part 3, Chapter 2, 2.5 2.5 Testing (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5>)
- Part 3, Chapter 2, 2.5.3 2.5.3 Test fixtures (<http://www.eac.gov/vvsg/part3/chapter02.php#2.5.3>)
- Part 3, Chapter 5, 5.3.3-A 5.3.3-A Reliability, pertinent tests (<http://www.eac.gov/vvsg/part3/chapter05.php#5.3.3-A>)
- Part 3, Chapter 5, 5.3.4-A 5.3.4-A Accuracy, pertinent tests (<http://www.eac.gov/vvsg/part3/chapter05.php#5.3.4-A>)

## Testing Lab

This term appears in:

- Introduction, Chapter 2, 2.1.1 2.1.1 VVSG Standards Architecture (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.1.1>)
- Introduction, Chapter 3, 3.3 3.3 The 2002 VSS (<http://www.eac.gov/vvsg/introduction/chapter03.php#3.3>)
- Introduction, Chapter 4, 4.3 4.3 Navigating Through Requirements (<http://www.eac.gov/vvsg/introduction/chapter04.php#4.3>)
- Part 1, Chapter 1, 1 1 Introduction (<http://www.eac.gov/vvsg/part1/chapter01.php#1>)
- Part 1, Chapter 2, 2.4-A 2.4-A Implementation statement (<http://www.eac.gov/vvsg/part1/chapter02.php#2.4-A>)
- Part 1, Chapter 3, 3.2.8.2-A 3.2.8.2-A Safety certification (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.8.2-A>)
- Part 1, Chapter 5, 5.8.8-A 5.8.8-A Physical encasing lock access requirement (<http://www.eac.gov/vvsg/part1/chapter05.php#5.8.8-A>)
- Part 1, Chapter 6, 6.4.5 6.4.5 Maintainability (<http://www.eac.gov/vvsg/part1/chapter06.php#6.4.5>)
- Part 2, Chapter 3, 3.2-A 3.2-A TDP, implementation statement (<http://www.eac.gov/vvsg/part2/chapter03.php#3.2-A>)
- Part 2, Chapter 3, 3.5.5-C 3.5.5-C TDP, physical lock documentation of use (<http://www.eac.gov/vvsg/part2/chapter03.php#3.5.5-C>)
- Part 3, Chapter 2, 2.4.1 2.4.1 Initiation of testing (<http://www.eac.gov/vvsg/part3/chapter02.php#2.4.1>)
- Part 3, Chapter 5, 5.1.1.1 5.1.1.1 Steady-state conditions (<http://www.eac.gov/vvsg/part3/chapter05.php#5.1.1.1>)
- Part 3, Chapter 5, 5.4.2-A 5.4.2-A OEVT team resources (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.2-A>)
- Part 3, Chapter 5, 5.4.6-A 5.4.6-A VSTL Response to OEVT (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.6-A>)

## Testing Phase

This term appears in:

- Part 1, Chapter 6, 6.4.1.8-G 6.4.1.8-G Do not disable error checks (<http://www.eac.gov/vvsg/part1/chapter06.php#6.4.1.8-G>)

## Usability Testing

This term appears in:

- Introduction, Chapter 2, 2.2 2.2 Usability Performance Requirements (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.2>)
- Introduction, Chapter 2, 2.3 2.3 Expanded Usability and Accessibility Coverage

- (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.3>)
- Introduction, Chapter 3, 3.4 3.4 HAVA and VVSG 2005 (<http://www.eac.gov/vvsg/introduction/chapter03.php#3.4>)
- Part 1, Chapter 1, 1.1.2 1.1.2 Usability Performance Benchmarks (<http://www.eac.gov/vvsg/part1/chapter01.php#1.1.2>)
- Part 1, Chapter 3, 3.1.2 3.1.2 Special terminology (<http://www.eac.gov/vvsg/part1/chapter03.php#3.1.2>)
- Part 1, Chapter 3, 3.2.1.2-A 3.2.1.2-A Usability testing by manufacturer for general population (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.1.2-A>)
- Part 1, Chapter 3, 3.2.7-A 3.2.7-A General support for alternative languages (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.7-A>)
- Part 1, Chapter 3, 3.2.8.1-B 3.2.8.1-B Usability testing by manufacturer for poll workers (<http://www.eac.gov/vvsg/part1/chapter03.php#3.2.8.1-B>)
- Part 1, Chapter 3, 3.3.2-A 3.3.2-A Usability testing by manufacturer for voters with low vision (<http://www.eac.gov/vvsg/part1/chapter03.php#3.3.2-A>)
- Part 1, Chapter 3, 3.3.3-A 3.3.3-A Usability testing by manufacturer for blind voters (<http://www.eac.gov/vvsg/part1/chapter03.php#3.3.3-A>)
- Part 1, Chapter 3, 3.3.4-A 3.3.4-A Usability testing by manufacturer for voters with dexterity disabilities (<http://www.eac.gov/vvsg/part1/chapter03.php#3.3.4-A>)
- Part 2, Chapter 3, 3.6.2 3.6.2 System test specifications (<http://www.eac.gov/vvsg/part2/chapter03.php#3.6.2>)
- Part 3, Chapter 2, 2.2 2.2 Scope of Assessment (<http://www.eac.gov/vvsg/part3/chapter02.php#2.2>)
- Part 3, Chapter 3, 3.3 3.3 Performance Testing (Benchmarking) (<http://www.eac.gov/vvsg/part3/chapter03.php#3.3>)
- Part 3, Chapter 5, 5.4.2-F 5.4.2-F OEVT level of effort – test plan (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.2-F>)

## Verification

Care must be taken in the definition of this term since no testing can establish the correct operation of a software system. This problem is equivalent to the Turing Halting Problem. Testing can only establish that a system does *not* work correctly and therefore, like the validation of scientific theory, the hypothesis that a system is correct must stand the test of time during which all assaults upon the system in terms of tests have failed to establish that the system works incorrectly.

This term appears in at least 120 sections of the VVSG.

## Vulnerability Testing

This term is defined internally in the VVSG in Part3, Chapter 3, 3.4 Vulnerability Testing as:

"Vulnerability testing is an attempt to bypass or break the security of a system or a device. Like functional testing, vulnerability testing can falsify a general assertion (namely, that the system or device is secure) but it cannot verify the security (show that the system or device is secure in all cases). Vulnerability testing is also referred to as penetration testing. Vulnerability testing can be performed using a test suite or it can be open-ended. Vulnerability testing involves the testing of a system or device using the experience and expertise of the tester; using the knowledge of system or device design and implementation; using the publicly available knowledge base of vulnerabilities in the system or device; using the publicly available knowledge base of vulnerabilities in similar system or device; using the publicly available knowledge base of vulnerabilities in similar and related technologies; and using the publicly available knowledge base of vulnerabilities generally found in hardware and software (e.g., buffer overflow, memory leaks, etc.)."

A definition for this term should also appear in the glossary.

This term appears in:

- Introduction, Chapter 2, 2.2 Introduction to New and Expanded Material (<http://www.eac.gov/vvsg/introduction/chapter02.php#2>)
- Introduction, Chapter 2, 2.5 2.5 Open-Ended Vulnerability Testing (<http://www.eac.gov/vvsg/introduction/chapter02.php#2.5>)
- Part 1, Chapter 7, 7.2-D 7.2-D EMS, ballot style protection (<http://www.eac.gov/vvsg/part1/chapter07.php#7.2-D>)
- Part 1, Chapter 7, 7.5.1.2-A 7.5.1.2-A Activation device, ballot secrecy (<http://www.eac.gov/vvsg/part1/chapter07.php#7.5.1.2-A>)
- Part 1, Chapter 7, 7.5.1.3-A 7.5.1.3-A Activation device, credentials and tokens (<http://www.eac.gov/vvsg/part1/chapter07.php#7.5.1.3-A>)
- Part 1, Chapter 7, 7.5.1.4-A 7.5.1.4-A Activation device, may access remote registration database

- (<http://www.eac.gov/vvsg/part1/chapter07.php#7.5.1.4-A>)
- Part 1, Chapter 7, 7.5.1.4-B 7.5.1.4-B Activation device, source code reviews (<http://www.eac.gov/vvsg/part1/chapter07.php#7.5.1.4-B>)
- Part 1, Chapter 7, 7.5.4-D 7.5.4-D DRE, cast is committed (<http://www.eac.gov/vvsg/part1/chapter07.php#7.5.4-D>)
- Part 1, Chapter 7, 7.6-A 7.6-A DRE, no CVRs before close of polls (<http://www.eac.gov/vvsg/part1/chapter07.php#7.6-A>)
- Part 1, Chapter 7, 7.6-B 7.6-B Programmed vote-capture devices, poll-closing function (<http://www.eac.gov/vvsg/part1/chapter07.php#7.6-B>)
- Part 1, Chapter 7, 7.8.3.1-F 7.8.3.1-F Precinct tabulators, no tallies before close of polls (<http://www.eac.gov/vvsg/part1/chapter07.php#7.8.3.1-F>)
- Part 3, Chapter 1, 1.1.4.3 1.1.4.3 Open-ended vulnerability testing (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.4.3>)
- Part 3, Chapter 2, 2.2 2.2 Scope of Assessment (<http://www.eac.gov/vvsg/part3/chapter02.php#2.2>)
- Part 3, Chapter 3, 3.4 3.4 Vulnerability Testing (<http://www.eac.gov/vvsg/part3/chapter03.php#3.4>)
- Part 3, Chapter 5, 5.4 5.4 Open-Ended Vulnerability Testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4>)
- Part 3, Chapter 5, 5.4.1-A 5.4.1-A Scope of open-ended vulnerability testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.1-A>)
- Part 3, Chapter 5, 5.4.1-B 5.4.1-B Focus of open-ended vulnerability testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.1-B>)
- Part 3, Chapter 5, 5.4.1-C 5.4.1-C OEVT General Priorities (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.1-C>)
- Part 3, Chapter 5, 5.4.2-B 5.4.2-B Open-ended vulnerability team establishment (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.2-B>)
- Part 3, Chapter 5, 5.4.2-F 5.4.2-F OEVT level of effort – test plan (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.2-F>)
- Part 3, Chapter 5, 5.4.3-A 5.4.3-A Rules of engagement – context of testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.3-A>)
- Part 3, Chapter 5, 5.4.4-A 5.4.4-A OEVT fail criteria – violation of requirements (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.4-A>)
- Part 3, Chapter 5, 5.4.4-B 5.4.4-B Threat model - failure (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.4-B>)
- Part 3, Chapter 5, 5.4.4-C 5.4.4-C OEVT fail criteria – critical flaws (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.4-C>)
- Part 3, Chapter 5, 5.4.6-A 5.4.6-A VSTL Response to OEVT (<http://www.eac.gov/vvsg/part3/chapter05.php#5.4.6-A>)

## White-Box

This term is erroneously defined identically to "Black-Box": "Testing technique focusing on testing functional requirements, those requirements being defined in an explicit specification. It treats the item being tested as a "black box," with no examination being made of the internal structure or workings of the item.'

This term appears in:

- Part 3, Chapter 1, 1.1.2 1.1.2 Applicability to COTS and borderline COTS products (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.2>)
- Part 3, Chapter 1, 1.1.3.2 1.1.3.2 Logic verification (<http://www.eac.gov/vvsg/part3/chapter01.php#1.1.3.2>)
- Part 3, Chapter 3, 3.2 3.2 Functional Testing (<http://www.eac.gov/vvsg/part3/chapter03.php#3.2>)
- Part 3, Chapter 5, 5.2.2-C 5.2.2-C Pass criteria for structural testing (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.2-C>)
- Part 3, Chapter 5, 5.2.2 5.2.2 Structural coverage (white-box testing) (<http://www.eac.gov/vvsg/part3/chapter05.php#5.2.2>)

Retrieved from

"[http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments\\_on\\_Appendix\\_A:\\_Definitions\\_of\\_Words\\_with\\_Spec](http://www.associationforsoftwaretesting.org/wiki/index.php?title=Comments_on_Appendix_A:_Definitions_of_Words_with_Spec)

Category: Discussion

- 
- This page was last modified 17:07, 23 March 2008.