Sub-agenda

1	2	3	
✓	○	○	Reliability and accuracy
○	○	○	COTS
○	○	○	Conformity assessment, scope of testing
○	○	○	Coding conventions and logic verification
○	○	○	California Volume Reliability Testing Protocol
○	○	○	_____ Write-in
○	○	○	_____ Write-in
○	○	○	_____ Write-in

Core Requirements and Testing

Reliability and accuracy:
Benchmarks, metrics, and test methods

David Flater

Computer Scientist

Order of presentation

- Background
- New reliability benchmark
- New reliability test protocol
- Accuracy

Background

Benchmark

- Definition: Quantitative point of reference to which the measured performance of a system or device may be compared
- Plain language: The number specified in the requirement (e.g., 163 hours)

Metrics

- Accuracy:
  - Ballot position error rate = # errors / # ballot positions = *n*
- Reliability:
  - MTBF = Time / # failures = *n* hours
  - Failure rate = # failures / time = *n* / hour
- Ballot misfeed rate = # misfeeds / # ballots = *n*

Test methods

- Errors, failures, misfeeds, ballot positions, ballots, and time can be tracked simultaneously during the execution of any test
- Given those data and the specified benchmarks, we must specify protocols for deciding acceptance and rejection of voting systems
- Usually, operational testing can only demonstrate nonconformity
- Current protocols for reliability and accuracy do demonstrate conformity, but there are compromises

The big picture

- In most cases, conformity to strict benchmarks cannot practically be demonstrated through operational testing
- Lax benchmarks do no one any good
- In other industries, performance to strict benchmarks is achieved through the application of available methods for design, quality assurance, and performance monitoring
  - Functional failure analysis
  - Operational testing is only a validation of the design
- See Max Etschmaier, “Voting machines: reliability requirements, metrics, and certification”

The little picture

- In the current Guidelines, the difficulty of demonstrating conformity is resolved in two ways
  - Compromised testing
  - Lax benchmarks

Compromised testing

- VVSG'05 permits test labs to bypass portions of the system that would be exercised during an actual election (VVSG'05 II.1.8.2.3)
  - "May use a simulation device… provided that the simulation covers all voting data detection and control paths that are used in casting an actual ballot."
  - But… "In the event that only partial simulation is achieved, then an independent method and test procedure shall be used to validate the proper operation of those portions of the system not tested by the simulator."
  - Also: "For systems that use a light source as a means of detecting voter selections, the generation of a suitable optical signal by an external device is acceptable."
  - Invalid system test

Lax benchmarks

- Product standard requirement: "The MTBF demonstrated during certification testing shall be at least 163 hours." (VVSG'05 I.4.3.3)
- The specified testing (VVSG'05 II.4.7.1, II.C.4) does not do that
  - MTBF demonstrated ranges from 44 hours to 73 hours (at 90 % confidence)
  - Minimum duration of test is 169 hours (was 163)

MTBF benchmarks in VVSG'05

Benchmark	P(failure) in 15 hours	Min. test time (if benchmark used as lower test MTBF)	w/ 7.1 days
45 hours (VVSG'05 II.C.4)	28 %	169 hours (7.04 days)	1 device
135 hours (VVSG'05 II.C.4)	11 %	21.1 days	3 devices
163 hours (VVSG'05 I.4.3.3)	8.8 %	25.5 days	4 devices

Specific advice received

Benchmark	From	P(failure) in 15 hours	Min. test	w/ 7 days	w/ 100 devices
1500 hours	[1]	1.0 %	234 days	34 devices	2.34 days
5000 hours	[2]	0.30 %	2.14 years	112 devices	7.82 days
15000 hours	[3]	0.10 %	6.42 years	335 devices	23.4 days

[1] IEEE Draft 5.3.1 ballot comment (later increased to 15000)
[2] VVSG'05 public comment #2056, lower bound
[3] VVSG'05 public comment #2056, upper bound

What's realistic (without simulation)?

- Actually been done: California Volume Reliability Testing Protocol for DREs
  - 100 devices
  - 50 "voters"
  - 6 hours
  - 110 ballots per device
  - Accept if ≤ 3 % of machines suffer "substantive failures"

Good news

- CA test uses real people interacting with the equipment in the way it is intended to be used
- Has proven itself worthy

Bad news

- 600 hours with 3 failures demonstrates MTBF ≥ 89.8 hours (at 90 % confidence)
  - P(failure) in 15 hours = 15 %
- With 0 failures it would be 260.6 hours
  - P(failure) in 15 hours = 5.6 %

What can be done

- We can improve the impact of testing without making it impossibly long
  - Use realistic volume test
  - Eliminate lax benchmarks
  - Lessen or eliminate tolerance of failures that occur during testing
  - Make full use of data collected throughout entire testing campaign
- However, the underlying problems will remain
  - Quality cannot be tested in; it must be built in

New reliability benchmark

What kind of benchmark do we want?

- A volume-based benchmark would be better
  - Time-dependent: continuously moving parts, perishable substances
  - Volume-dependent: parts that move when something happens, certain software failures (e.g., crashes due to memory leaks)
  - MTBF says nothing about the workload
  - Equipment is unlikely to fail with no workload

Volume means different things

- Vote-capture devices and optical scanners get volume as # ballots, # ballot positions, # votes, …
- But not everything does
  - Central EMS that just consolidates input from the precincts
  - Smart card activators

Good news

- We can specify different benchmarks for different types of devices according to whatever concept of volume or time is most appropriate to them
- To the test method, a ratio is a ratio

What is the new number?

- For each type of device, you decide:
  - What proportion you are willing to have fail during an election (*p*)
  - The time, # ballots, or whatever (*w)* per device for an election
  - Relevant assumptions
- Acceptable failure rate = $-\ln(1-p) / w$
  - For p = 3 %, w = 110 ballots, the acceptable ballot failure rate = 2.77×10^{-4} (1 failure per 3610 ballots)
  - For p = 3 %, w = 6 hours, the acceptable failure rate = 5.08×10^{-3} / hour (MTBF = 197 hours)
- If you only want one benchmark, then choose *p* and *w* to be applicable for all devices

New reliability test protocol

Issues with current test protocol

- Simulated volume
- High tolerance for failures observed during test
  - Need at least 2 failures to reject
  - Up to 6 failures are tolerated
- Assumes a self-contained test
  - No protocol for failures occurring during other testing
- Duration of testing driven by benchmark, test parameters

Changes discussed

- Collect data across all valid system tests
  - To include at least one good volume test
  - Ignore tests where simulation is required (e.g., for safety of personnel) or errors are forced
- Reject if data collected show with 90 % confidence that the system does *not* conform
  - Even on a single failure

Changes discussed

- Report the performance that was demonstrated with 90 % confidence
  - Varies with length/volume of testing, number of failures observed, …
- Do not require exhaustive testing

Accuracy

Accuracy

- Ballot position error rate
- Not such a lax benchmark
- Limit = 1 / 500 000

Accuracy issues

- Simulated volume
- Ambiguous metric confuses ballot positions with votes (VVSG'05 II.C.5)
- Unclear how inaccuracies in ballot counts and totals of undervotes and overvotes factor in
  - Model assumes maximum one error per ballot position
- Untestable benchmarks on low-level operations

Changes discussed

- One end-to-end benchmark for testing
  - May retain benchmarks on low-level operations as design guidance, but not used in testing
- New metric: report total error rate
- Use same protocol as for reliability
  - Harmonize model
  - Harmonize to 90 % confidence needed for rejection

Report total error rate

- Need a definition of error that allows them to be counted
  - Errors must be observable
  - Given test report, tell me how many errors were made
- It's not as simple as the Guidelines imply

Report total error rate

(generalized from 1990 VSS F.6)

- **Report item:** Any one of the numeric values (totals or counts) that must appear in any of the vote data reports. Each ballot count and each vote, overvote and undervote total for each candidate or measure is a separate report item.
- **Report item error:** Absolute value of the difference between the correct value and the reported value.
- **Report total error:** Sum of all of the report item errors.
- **Report total volume:** Sum of all of the *correct* values.
- **Report total error rate** = report total error / report total volume

Unfinished business

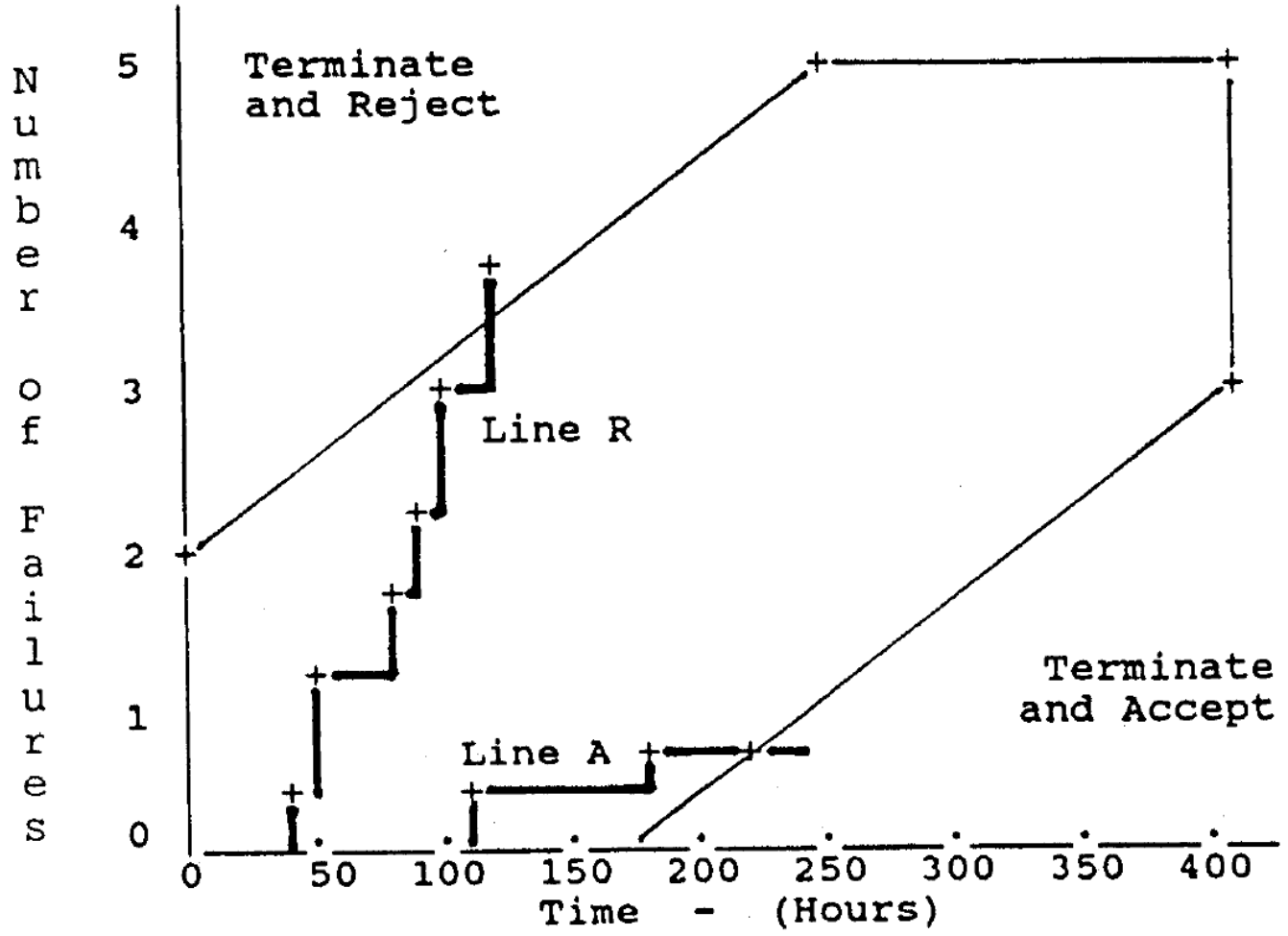
- Details, details: granularity of benchmarks
  - Reliability should be device-level
  - Accuracy should be system-level
- **Input needed from election officials**
  - Acceptable % failures: 0 % .. 30 %
  - Acceptable # errors: 0 .. 1000
  - Volumes for each type of device

Extra slides

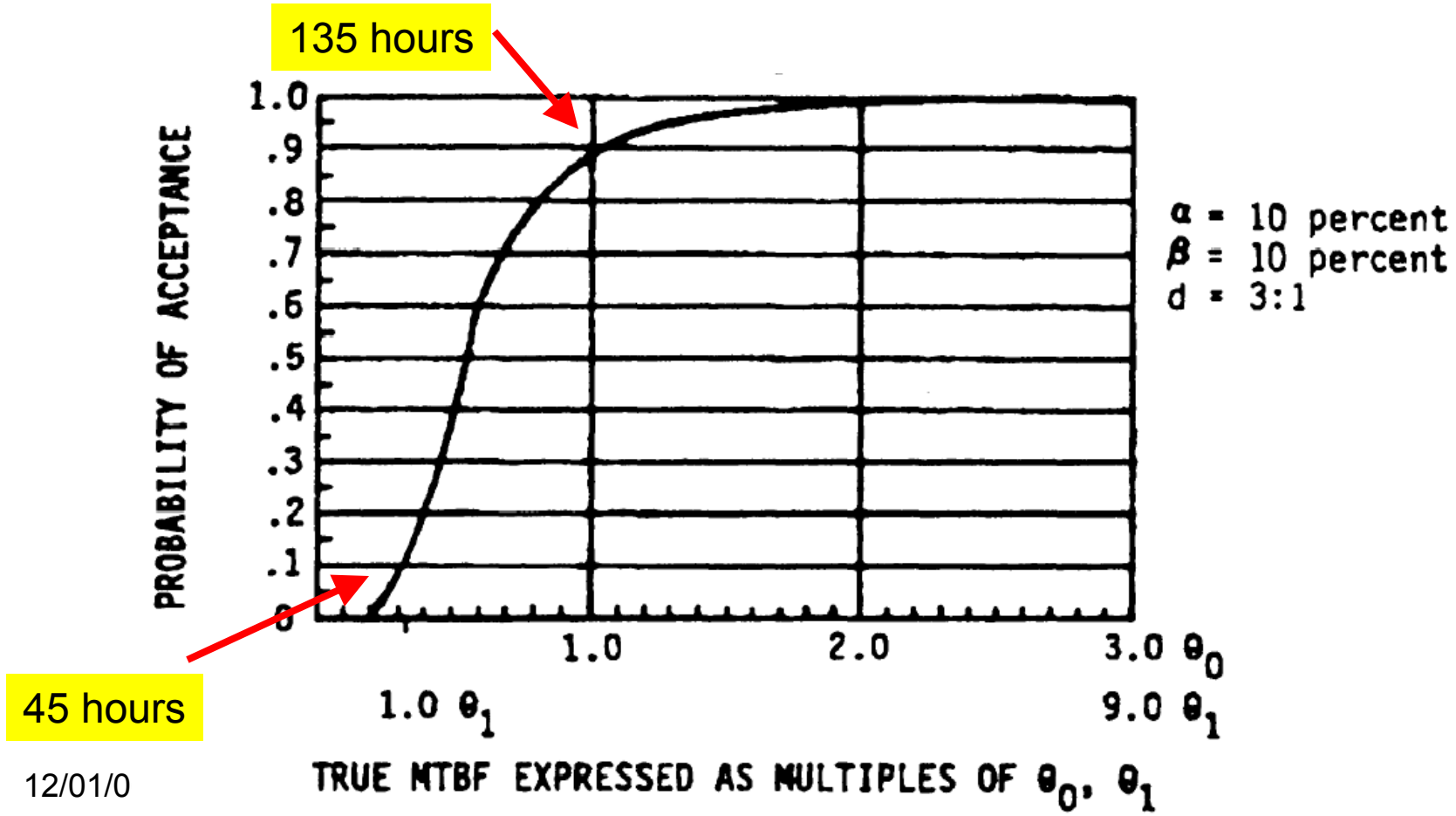
Compromised testing, motives

- Accuracy test requires 1 549 703 ballot positions—testing DREs requires too much typing
  - If the test is compromised, the number of ballot positions is of little relevance
  - “One flight test is worth a thousand simulations” – Henry Spencer
- To save time, want to run concurrent with temperature and power variation tests (VVSG’05 II.4.7.1); not practical to put people in test chamber the whole time
  - It would be better to collect data across all tests
  - The ones where we can’t do a complete system test should be the exceptions
  - Specifying one of these exceptions as *the* reliability, accuracy, etc. test is not preferable

Sequential test (example)



Current reliability test protocol



Whither 163 hours?

- In 1990 and 2002 VSS, minimum duration of Probability Ratio Sequential Test to demonstrate MTBF ≥ 45 hours with the given parameters was calculated to be 163 hours
  - In VVSG'05, this was revised to 169 hours
- MTBF actually demonstrated is yet another different number, and it varies
- Confusion of non-comparable values

Whence 45 hours?

- "A typical system operations scenario consists of approximately 45 hours of equipment operation, consisting of 30 hours of equipment set-up and readiness testing and 15 hours of elections operations." (VVSG'05 I.4.3.3)
- With MTBF = 45 hours, 63 % of our machines die by the end of the election (after 45 hours)
- More confusion of non-comparable values

Definition

- **failure:** (Voting system reliability) Event that results in (a) loss of one or more functions, (b) degradation of performance such that the device is unable to perform its intended function for longer than 10 seconds, (c) automatic reset, restart or reboot of the voting system, operating system or application software, (d) a requirement for an unanticipated intervention by a person in the role of poll worker or technician before the test can continue, or (e) error messages and/or audit log entries indicating that a failure has occurred. (Source: Expanded from 2002 VSS I.3.4.3.)

Probability of failure

- Calculate probability of at least one failure given MTBF θ (163 hours) and time t hours (15 hours)
- Wrong answer: p = t/θ = 9.2 %
  - Assumes uniform distribution
- Right answer: p = $1-e^{-t/\theta}$ = 8.8 %

Minimum duration of reliability test

- Given minimum acceptable MTBF θ_1 (45 h), calculate time to acceptance t_A assuming 0 failures
- Using MIL-HDBK-781A Test Plan V-D
  - Producer's risk = 11.1 % (nominally 10 %)
  - Consumer's risk = 10.9 % (nominally 10 %)
  - Discrimination ratio = 3
- Figure 13 (p. 227 in handbook): accept at $t \geq 3.75 \times \theta_1$
- For θ_1 = 45 h, t_A = 169 h
- Point at which 45 h has been demonstrated to 90 % confidence is actually somewhat earlier (104 h)
- Consumer's risk and confidence interval on demonstrated MTBF are not comparable values

MTBF demonstrated—recipe #1

- Using MIL-HDBK-781A Test Plan V-D, given minimum acceptable MTBF θ_1 and a test that ended with acceptance after *r* observed failures, calculate the MTBF that was demonstrated with 90 % confidence
- Table IXA (p. 115 of handbook; *c.f.* p. 43)

# failures	Total test time	Demonstrated MTBF
0	$3.75 \times \theta_1$	$1.6286 \times \theta_1$
1	$5.40 \times \theta_1$	$1.2950 \times \theta_1$
2	$7.05 \times \theta_1$	$1.1861 \times \theta_1$
3	$8.70 \times \theta_1$	$1.1357 \times \theta_1$
4	$10.35 \times \theta_1$	$1.1087 \times \theta_1$
5	$10.35 \times \theta_1$	$1.0481 \times \theta_1$
6	$10.35 \times \theta_1$	$.9811 \times \theta_1$

MTBF demonstrated—recipe #2

- Given observed MTBF θ after r failures, calculate the lower MTBF that was demonstrated with 90 % confidence
- Table XIV, p. 133 of handbook (*c.f.* p. 51 and Figure 22 on p. 240)
- To do it this way, need at least one failure…

# failures	Demonstrated MTBF
1	0.257 × θ
2	0.376 × θ
3	0.449 × θ

MTBF demonstrated—textbook approach

- Given test of duration t with 0 failures, calculate the MTBF θ that was demonstrated with 90 % confidence
- Probability that MTBF is > θ is equal to the probability of getting at least one failure if MTBF = θ
  - $0.9 = 1 - e^{-t/\theta}$
  - $\theta = t / \ln(10) = t \times 0.434294$
- For t = 168.75 hours (θ_1 = 45 hours)
  - This says 168.75 hours / ln(10) = 73.287 hours
  - Table IXA in handbook says 1.6286 × 45 hours = 73.287 hours
- With roundoff, t = 169 hours gives θ = 73.4 hours

MTBF demonstrated—textbook approach

- General case: given test of duration *t* with *r* failures, calculate the MTBF θ that was demonstrated with 90 % confidence

$$0.9 = 1 - \sum_{x=0}^{r} \frac{e^{-\frac{t}{\theta}} \left(\frac{t}{\theta}\right)^x}{x!}$$

- No closed-form solution; solve numerically
- For *t* = 600 hours and *r* = 3
  - This says θ = 89.8 hours
  - Table XIV in handbook says 0.449 × 200 hours = 89.8 hours

Limitations of testing

- Observed performance is not equivalent to true, average case performance
- Demonstrated performance is not equivalent to true, average case performance
- Consumer's risk: Probability of accepting a system with true performance equal to lower benchmark (worse)
- Producer's risk: Probability of rejecting a system with true performance equal to upper benchmark (better)
- Assumption that sample tested is representative

Accuracy protocol

Std	α	β	H_1	H_0	Reject on 1	Accept on 0	Accept on 1
1990	5 %	5 %	10^{-5}	10^{-7}	167 753	297 589	762 763
2002 2005	5 %	5 %	2×10^{-6}	10^{-7}	26 997	1 549 703	3 126 404

Alternatives to report total error rate

- Like with reliability, you decide the “whatever” (*w*)
- …but I also need a viable definition of error that makes them countable
- One bad report = one error?
  - Assumes that being off by 1 is as bad as being off by 1 000 000
  - Assumes that being off by 1 000 000 is no worse than being off by 1

Point/counterpoint

- “Collect data across all tests” considered harmful—data collected from volume test (simulating election conditions) is more credible than data collected in other tests (possibly not representative)
- Volume test is still a relatively predictable and repetitive workload; variety of workload is more credible, even if somewhat “lab-flavored”

Point/counterpoint

- Must demonstrate conformity or there is no standard—lower benchmark if necessary
- With a lax benchmark, you never fail anybody; with a strict benchmark, at least you can fail those that demonstrate nonconformity

Point/counterpoint

- Availability requirement is needed
- Maintenance during election generates accusations of tampering, so time to repair is moot
- That's a complaint about the procedures; out of scope

Point/counterpoint

- Zero failures is unrealistic; must set benchmarks attainable by current systems
- Much complaining about current systems

Point/counterpoint

- “Pilot error” should still count as a failure
- Usability, reliability, and test lab error are all separate concerns

Core Requirements and Testing

COTS

David Flater
Computer Scientist

Resolution #14-05

- Motivations
  - Belief that COTS is exempt from testing
  - “Rotten COTS” not suitable for voting use
- Reality check
  - COTS is not exempt, but the Guidelines are confusing
  - Current terminology (“COTS” vs. “non-COTS”) is inadequate to clarify all cases
  - Reinventing the wheel *is* bad… unless it was a rotten design

Levels of scrutiny

Categories	Level of scrutiny	Tested?	Source code/data required?	Coding standards enforced?	Shown to be correct?
COTS	Black box	Yes	No	No	No
Third-party logic, border logic, configuration data	Clear box	Yes	Yes	No	No
Application logic	Coding standards	Yes	Yes	Yes	No
Core logic	Logic verification	Yes	Yes	Yes	Yes

Black box testing

- **COTS:** Software, firmware, device or component that is used in the United States by many different people or organizations for many different applications and that is incorporated into the voting system with no vendor- or application-specific modification.

Coding standards

- **Application logic:** Software, firmware, or hardwired logic from any source that is specific to the voting system, with the exception of border logic.

Logic verification

- **Core logic:** Subset of application logic that is responsible for vote recording and tabulation.

Clear box testing

- **Border logic:** Software, firmware, or hardwired logic that is developed to connect application logic to COTS or third-party logic.
- **Configuration data:** Non-executable input to software, firmware, or hardwired logic.
- **Third-party logic:** Software, firmware, or hardwired logic that is neither application logic nor COTS; e.g., general-purpose software developed by a third party that is either customized (e.g., ported to a new platform, as is Windows CE*) or not widely used, or code generated by a COTS package.

* Commercial equipment and materials are identified in order to describe certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

"Rotten COTS"

- COTS is not automatically excluded from anything except the requirement to deliver source code
- The test lab must make a determination whether previous certifications and field experience render any portion of the test campaign redundant
- This is not a COTS-specific issue
  - E.g., hardware already certified as FCC Class A, COTS or otherwise
  - E.g., modifications to previously certified systems
- Any reduction in the scope of testing must be justified in the test plan and approved by the EAC
- "Rotten COTS" will not qualify for a reduced scope of testing

COTS authentication

- Is what the vendor represented as COTS really COTS?
- Solution: Test lab obtains COTS independently
- Integrate or witness integration into equipment to be tested

Unfinished business

- Inconclusive discussion on STS toward providing a more precise definition of COTS
  - Publicly available
  - Widespread use (≥ 10 000 instances)*
  - Maintainer has existed for 7 years*
  - Proper configuration management
- * These criteria have unresolved issues

EAC opportunity

- Maintain list of COTS products previously found to have been acceptable for use in voting systems
- Input into determination of test plan
- Test lab must ensure that the use of the COTS product in a new system is comparable to its use in the previously approved system
- No waiver from system testing

Core Requirements and Testing

Conformity assessment, scope of testing

David Flater
Computer Scientist

Conformity assessment

- Adherence of product to requirements in the Guidelines
- Anything not specified in the Guidelines is irrelevant unless it is required to test things that *are* specified
- Strive for maximum objectivity
- Repeatability and reproducibility
- Rejection must be defensible in terms of unmet requirements

Issues

- The VVSG'05 testing volume requires testing of vendor-specific functionality (VVSG'05 II.3.2.3, II.6.3, II.6.7)
  - This is not conformity assessment
  - Not traceable to requirements of Volume I
- Pressure for federal testing to do more for the states
- Impossible to include all state-specific requirements in federal standard

Discussion

- The Testing Standard of the 2007 Voluntary Voting System Guidelines shall not require the test lab to perform activities beyond the scope of assessing conformity to the Guidelines
- This does not preclude the EAC from adding requirements and/or criteria beyond the VVSG for certification, nor does it preclude test labs from performing additional tests

Core Requirements and Testing

Coding conventions and logic verification

David Flater
Computer Scientist

Preface

- The directions taken on coding conventions and logic verification and much of the presentation material are unchanged since they received general approval at the 2005-09 TGDC meeting

What are coding conventions?

- Mostly, requirements on the *form* (not *function*) of source code
- Some requirements affecting software integrity, implemented as defensive coding practices
  - Error checking
  - Exception handling
  - Prohibit practices that are known risk factors for latent software faults and unverifiable code

Current direction (approved 2005-09)

- Expand coding conventions addressing software integrity (the 20 % with 80 % impact)
  - Start with IEEE requirements
  - Make defensive coding requirements more explicit
  - Require block-structured exception handling
- Clarify length limits (modules vs. callable units)
- Delete the 80 % with only 20 % impact; require use of “published, credible” coding conventions instead

EAC opportunity

- Periodically review current best practices (yearly)
- Publish list of coding conventions acceptable for use in voting systems
- Eliminates vague definition of “credible” coding conventions

Anticipated controversy

- Gored ox: C doesn't have block-structured exception handling
- Block-structured exception handling is now accepted as part of structured programming
- VVSG require behaviors that are representative of block-structured exception handling
- Three migration paths for legacy C code
  - Java (**1995**)
  - C++ (ISO/IEC 14882, **1998**)
  - C# (ISO/IEC 23270, **2003**)
- Alternatives w/o migration
  - Ada (ANSI/MIL-STD-1815A, **1983**)
  - Visual Basic .NET (**2002**)
  - Many others

The case for exceptions

- "One of the major difficulties of conventional defensive programming is that the fault tolerance actions are inseparably bound in with the normal processing which the design is to provide. This can significantly increase design complexity and, consequently, can compromise the reliability and maintainability of the software."
  - M. R. Moulding, "Designing for high integrity: the software fault tolerance approach," Section 3.4. In C. T. Sennett, ed., High-Integrity Software, Plenum Press, New York and London, 1989.

What is logic verification?

- Formal characterization of software behavior within a carefully restricted scope
- Proof that this behavior conforms to specified assertions (i.e., votes are reported correctly in all cases)
- Complements [falsification] testing
- C.f. “inductive assertions,” “Hoare logic,” “program proving”

Motivation

- TGDC Resolution #29-05, “Ensuring Correctness of Software Code”
- Higher level of assurance than operational testing alone
- Clarify objectives of source code review

How it works

- Vendor specifies pre- and post-conditions for each callable unit
- Vendor proves assertions regarding tabulation correctness
- Testing authority reviews, checks the math, and issues findings
  - Pre- and post-conditions correctly characterize the software
  - The assertions are satisfied

Compromise #1

- Scope of verification limited to core logic
- **Core logic:** Subset of application logic that is responsible for vote recording and tabulation.
- Limited scope = limited assurance; unlimited scope = impracticable

Compromise #2

- Programming language does not have formally specified semantics
- A formal proof cannot be mandated
- Do what is feasible
  - Formality where possible
  - Informal arguments where not
  - Limitations on complexity to make correctness intuitively obvious
- Still better than operational testing alone

Potential criticisms

- Too rigorous
- Not rigorous enough
- Too complicated
- Oversimplified
- Only appropriate for safety-critical systems
- Mockery of what is done for safety-critical systems
- …

Extra slides

Coding conventions motivation

- Started in 1990 VSS, expanded in 2002, expanded more in IEEE P1583 5.3.2b
- TGDC Resolution #29-05, “Ensuring Correctness of Software Code” (part 2)
- Enhance workmanship, security, integrity, testability, and maintainability of applications

2002 VSS / VVSG'05 status

- Mixture of mandatory and optional
- Vendors may substitute "published, reviewed, and industry-accepted coding conventions"
- Incorporated conventions suffered from rapid obsolescence and limited applicability
- Some mandatory requirements had unintended consequences

Credible ≈ industry-accepted

- Coding conventions shall be considered credible if and only if at least two different organizations with no ties to the creator of the rules or to the vendor seeking certification, and which are not themselves voting equipment vendors, independently decided to adopt them and made active use of them at some time within the three years before certification was first sought.

Coding conventions non-issues

- Assembly language in “hardware-related segments” and operating system software
  - It’s been handled; see COTS presentation
- Grandfathering of stable code
  - Coding conventions: 3-year rule relative to first certification
  - In general, grandfathering is states’ prerogative
- COTS or “slightly modified” COTS
  - It’s been handled; see COTS presentation

Logic verification non-issue

- Rice's theorem: In the general case, nontrivial properties are undecidable
- *This is not the general case*
- Vote counting uses very simple math and logic
- All voting system designs must preserve the ability to demonstrate that votes will be counted correctly
- Would you want one that didn't?

Core Requirements and Testing

California Volume Reliability Testing Protocol

David Flater
Computer Scientist

In a nutshell

- Part of California certification testing
- “Conditions approximating normal use by voters in a polling place on Election Day”
- Hire “extras” to act as voters
- Casting, counting, reporting

Parameters

- For DREs:
  - 100 devices
  - 50 “voters”
  - 6 hours
  - 110 ballots per device
  - True end-to-end system test

Parameters

- For Precinct Count Optical Scanners:
  - 50 devices
  - 10 “voters”
  - 400 ballots per device
  - Test decks provided by vendor…
    - Fair enough for reliability testing (the nominal purpose)
    - For accuracy testing, would like a realistic range of marks
    - No good for usability testing

Needed for VVSG'07

- Benchmarks
  - Reliability
  - Accuracy
  - Rate of misfeeds
  - Usability
- Need a credible volume test
- Simulation considered harmful

Cost

- This is not what the test labs do now
- More devices + more people = more $
- But not doing it would be hard to defend