

5 Basic Rules for Testing Your Voting Machinery

By
John Washburn,
Certified Software Quality Engineer

With the introduction of new voting systems there are a number of experienced election officials who will need to effectively test software in the new voting machinery they have purchased. Also new clerks are elected to office, members of the public are now interested in observing the testing and certification of voting machinery, and members of the media are also interested in understanding how voting machine software should be tested.

This summary is directed to all people interested in performing or observing the testing of the software of an election system. Here are the 5 basic rules for testing election system software.

1. Create your own test deck.

Always create the test plan and test ballots yourself. Do not accept test materials from the vendor or programmer.

2. Always test an election system in "Election Mode."

Many elections systems offer "test mode," "automatic testing," "self-testing," or other such features. Never test an election system in these test modes. Since the election official performing the testing is signing paperwork stating the election system will work during an election, the testing **MUST** be done in the same mode used during the election. Otherwise, the election official has no knowledge of how it will perform in an election.

3. Create a comprehensive deck of test ballots.

The goal of L&A testing is to determine if the software is or is not working as expected. Thus, the deck of test ballots for L&A testing must be **designed** to actively look for programming defects. A haphazard collection of test ballots will not do. A comprehensive deck of test ballots must be thoughtfully **constructed**. For voting machines there are 5 general steps to creating a deck of good test ballots for L&A testing

- a. Create 2 ballots with zero marks (no screen touches). This tests that only designated areas are read.
- b. Create every possible ballot which has exactly 1 mark (one screen touch). This tests the calibration.
- c. Create every possible ballot which has exactly 2 marks (two screen touches). This test for overvotes.
- d. If there is a race where the elector votes for N of M candidates, then, for only the multi-vote race(s), create every possible ballot which has exactly 3 marks (screen touches), 4 marks (screen touches), 5 marks (screen touches) etc., up to a limit of ballots with exactly N+1 marks (screen touches).
- e. Create tally distinction ballots such that every ballot line on the ballot will have a distinct expected vote total. This tests ballot rotation and that candidate totals are not swapped.

For complete instructions, see [Guidelines to Create Deck of Test Ballots¹](#).

4. Perform the Planned Tests.

Often software tests are planned but never executed. There are many reasons, but the simplest is the lack of time. Even if time becomes short, perform the tests you have planned. The test plan was a good idea when time permitted. It is even better when the schedule crush begins. The worst time to decide which corners to cut and which tests to skip is under the pressure of looming deadlines.

5. Go wherever the evidence leads.

The software testing may well discover defects. There may be tremendous pressure to ignore the troublesome results or pressure to allow an illegal software patch to "fix" the discovered problem. Resist this pressure. The uncomfortable answers from the software testing results may mean the election system under test is unfit or illegal to use in the upcoming election, or the system can only be used if new procedures or protocols are followed.

¹ <http://www.washburnresearch.org/archive/TestingGuidelines/GuidelinesForCreatingTestBallots.pdf>