Perfect Software. . . and other illusions about testing

Gerald M. Weinberg

1 Why Do We Bother Testing

- Striving for perfection
- Not making decisions
- Not recognizing all the information needed for decision-making
- Putting wrong priorities on various risks
- Believing testing can improve a product
- Believing there's a "'testing phase"' during which all testing – and only testing – is done

2 What Testing Cannot Do

- Not honoring testers
- Over-honoring testers
- Scapegoating testers
- Not using the information gleaned from testing or other sources
- Making decisions that are emotional, not rational
- Not evaluating the quality of test data
- Testing without adequate preparation
- Failing to coordinate testing with the rest of a project
- Rushing the teters

- Not insisting on due diligence from managers
- Assuming that others' decisions are not rational just because the don't agree with yours
- Not realizing there is more than one use for information from testing

3 Why Not Just Test Everything?

- Demanding "'test everything"'
- Not understanding sampling
- Spending too much for information that's not worth it
- Testing for the sake of appearance
- Not using all sources of information
- Thinking that machines can perform exhaustive testing, even if people can't
- Increasing risk by constraining resources

4 What's the Difference Between Testing and Debugging?

- Thinking that locating errors can be scheduled
- Not considering time lost to task-switching
- Treating testing as a low-priority task that can be interrupted for just about any reason
- Demanding that testers pinpoint every failure
- Demanding that testers locate every fault
- Repairing without retesting
- Ignoring cross-connections
- Paying insufficient attention to testability
- Insisting that all bugs be "'reproducible"'
- Confusing testing with "'creating and executing test cases"'
- Demanding process overhaul in your company

5 Meta-Testing

- Believing that all relevant information is contained in test reports
- Believing you can sit in your office and know what is going on with testing
- Believing that tests can "‘prove"’ anything correct
- Believing that the mere existence of documents has some value
- Allowing the list of bugs pending assessment/fixing/assignment to grow beyond human comprehension
- Blaming people so they feel motivated to hide bugs
- Rewarding people for going through the motions
- Not recording every identified failure
- Over-recording every identified failure
- Letting emotions determine what is tested and reported
- Using phony models to assess progress
- Assuming the official process description is always followed reliably and correctly
- Believing in objectivity
- Failing to review carefully any document produced using a template

6 Information Immunity

- Failing to notice when people are fearful
- Creating a fearful environment
- Allowing your fears to override the facts when making decisions
- Allowing your hopes to override facts when making decisions
- Indulging in compulsive behavior
- Assuming that any argument against your own point of view is part of a pathology
- Outright denial
- Thinking it can’t happen here

7 How to Deal With Defensive Reactions

- Failing to take differences into account
- Telling people they don't care about quality
- Leaving your brain outside
- Being overcritical of yourself
- Not being critical of yourself

8 What Makes a Good Test?

- Not thinking about what information you're after
- Measuring testers by how many bugs they find
- Believing you can know for sure how good a test is
- Failing to take context into account
- Testing without knowledge of the product's internal structure
- Testing with too much knowledge of the product's internal structure
- Giving statistical estimates of bugs as if the numbers were fixed, certain numbers
- Failing to apply measures of "'badness"' to your tests
- Not ensuring that development is done well
- Not considering the loss of testing efficiency caused by numerous found bugs

9 Major Fallacies About Testing

- Believing that blame works in the long run
- Believing that your first imrpression of a problem is always correct
- Believing that you can test anything "'exhaustively"'
- Thinking you can develop software "'quick and dirty"' and then test quality in
- Skipping unit testing as redundant because system testing will catch all the bugs
- Skipping system testing in the belief that it's redundant because unit testing will catch all the bugs
- Expecting testing to produce quality

10 Testing Is More Than Banging Keys

- Thinking a computer cann read minds
- Failing to verify software sales claims
- Failing to use coverage tools (for example, The White Glove Test) in your testing
- Thinking that coverage tests prove something is tested
- Confusing process documents with processes
- Confusing document with facts
- Failing to "‘eat your own dog food"’
- Using only non-representative "‘dogs"’ in a Dog Food Test
- Failing to test your testers, or testing them too much
- Pretending demonstrations are tests

11 Information Intake

- Not thinking about what information you’re after
- Not actively seeking the information you’re after
- Conflating intake and meaning
- Forbidding testers to look for bugs in certain places
- Failing to provide adequate equipment and tools for testing
- Succumbing to The Golden Elephant Syndrome

12 Making Meaning

- Jumping to conclusions about what data mean
- Running tests without documenting the expected results in advance
- Over-documenting expected results in advance
- Trying to make meaning all by yourself
- Thinking that meaning completely determines significance

13 Determining Significance

- Confusing repair difficulty with significancy
- Misjudging the significance of the speed of a response
- Failing to realize that significance is political
- Believing there is a "‘rational"’ or objective way to assess significance
- Allowing bullaby language to influence your assessment of significance
- Ignoring the significance of your actions on the project team itself

14 Making a Response

- Depending on luck
- Reducing test time and resources to make a schedule
- Failing to adjust schedules and estimates as testing provides information on the actual state of the product
- Failing to collect process data
- Not understanding when testing starts
- Testing a deas horse

15 Preventing Software Testing from Growing More Difficult

- Underestimating the complexity of old, patched-up code
- Not allowing these matters to be discussed, let alone measured
- Failing to adjust process data as current experience indicates
- Using early returns as indicator of later results
- Thinking about testers as "‘the bad guys who prevent delivery"’
- Testers thinking of themselves as "‘quality poilice"’

16 Testing Without Machinery

- Not recognizing the value of technical reviews as a complementary form of testing
- Falling for one of the many arguments for skipping technical reviews (or any part of your process, for that matter)
- Using technical reviews as punishment
- Skipping reviews to save time
- Failing to review designs and code for testability
- Failing to include tester as reviewers
- Failing to recognize the value of learning

17 Testing Scams

- Relying solely on numbers to manage a project
- Accepting testimonials through a third party

18 Oblivious Scams

- Using ealry reports of errors from a shipped product to estimate total errors shipped
- Making bug reporting tedious or inconvenient
- Creating a blaming environment that encourages falsified test reporting
- Rewarding form over context
- Rewarding quantitiy over quality