Measuring the Effectiveness of Software Testers

Quality Week 2002

Cem Kaner

Bottom-Line: My recommended approach

- My approach to evaluating testers is
  - Multidimensional
  - Qualitative
  - Multi-sourced
  - Based on multiple samples
  - Individually tailored
  - Intended for feedback or for determining layoff / keep, but not for quantifying the raise

Multidimensional

- If we measure on only one or a few dimensions, we will have serious measurement distortion problems, and probably measurement dysfunction.
- Common disasters:
  - Bug counts don't measure productivity, skill, or progress
  - Hours at work don't measure productivity, skill, or dedication
  - Certification doesn't measure productivity, skill, knowledge, competence, or professionalism
  - Peer ratings can easily degenerate into popularity contests
  - “Customer” ratings (e.g. ratings by programmers, etc.) can easily degenerate into popularity contests and foster gutlessness.

Robert Austin on Measurement Dysfunction

- An example of one-dimensional measurement: Schwab and U.S. Steel
  - Classic example: Schwab goes to the steel plant one day and marks the number of ingots of steel
  - Next day he comes back. His chalked number is crossed out and replaced with a larger one (more steel today)
  - Next day, even more steel (magic!)
  - The moral of the story (classically) is that things improve when you measure them.
  - Questions
    - *How might these people have improved <u>measured</u> productivity?*
    - *What side effects might there have been of improving <u>measured</u> productivity?*

Measurement Distortion and Dysfunction

- In an organizational context, dysfunction is defined as the consequences ***of organizational actions that interfere with the attainment of the spirit of stated intentions of the organization. (Austin, p. 10)***
- ***Dysfunction involves fulfilling the letter of stated intentions but violating the spirit.***
- ***A measurement system yields distortion if it creates incentives for the employee to allocate his time so as to make the measurements look better rather than to optimize for achieving the organization's actual goals for his work.***
- ***The system is dysfunctional if optimizing for measurement so distorts the employee's behavior that he provides less value to the organization than he would have provided in the absence of measurement.***

Austin on the 2-Party Model

- Principal (employer)
  - The person who wants the result and who directly profits from the result.
  - In the classic two-paty model, we assume that the employer is motivated by maximum return on investment
- Agent (employee)
  - In the classic two-party model, the employee wants to do the least work for the most money

Supervisory issues in the 2-party model

- No supervision
  - No work
- Partial supervision
  - Work only on what is measured
- Full supervision
  - Work according to production guidelines laid out by the employer
- There is no risk of distortion or dysfunction in the two-party model because the employee won't do anything that isn't measured. More measurement yields more work and more work, no matter how inefficient, yields more results than less work.

Austin's 3-party model

- Principal (Employer)
  - With respect to the employee, same as before: least pay for the most work.
  - With respect to the customer, wants to increase customer satisfaction
- Agent (Employee)
  - With respect to employer, same as before: least work for the most pay
  - With respect to the customer, motivated by customer satisfaction
- Customer
  - Wants the most benefit for the lowest price

Supervisory Issues in the 3-party model

- If there is NO SUPERVISION
  - Employee optimizes the time that she spends working, so that she provides the most customer benefit that she can, within the labor that he provides.
  - Employee works to the extent that increasing customer satisfaction (or her perception of customer satisfaction) provides more "benefit" to the employee than it costs her to work.
- If there could be FULL SUPERVISION
  - The employee would do exactly what the employer believed should be done to increase customer satisfaction.
  - Full supervision is infinitely expensive and, for other reasons as well, impossible. We are stuck with partial supervision.

Supervisory Issues in the 3-party model

- Effect of PARTIAL SUPERVISION
  - Employee is motivated by
    - increased customer satisfaction and by
    - rewards for performing along measured dimensions.
  - To the extent that the agent works in ways that don’t maximize customer satisfaction at a given level of effort, we have ***distortion***.
  - To the extent that the agent works in ways that reduce customer satisfaction below the level that would be achieved without supervision, we have ***dysfunction***

Austin's 3-Party Model

- A key aspect of this model is that it builds in the notion of internal motivation.
- Under ***full supervision*** with forcing contracts, perhaps internal motivation is unnecessary. (I disagree, but perhaps we can pretend that it is unnecessary.)
- Under ***partial supervision*** and ***no supervision***, internal motivation plays an important role in achieving customer satisfaction and in eliciting effort and results from the agent.
- This comes into play in Austin’s vision of delegatory management.

Multidimensional: So what should we measure?

- Obviously, we can't measure everything.
- I think that much of the distortion problem arises because the employee suspects the fairness of the measurement system.
- The important false simplification in the 2-party and 3-party model is that the employee is not motivated to please the employer.
  - If you (or your most visible corporate executive) are a jerk, you may succeed in so alienating your staff that you achieve the 2-party or 3-party model.
  - If you build credibility and trust with your staff, then increasing your actual satisfaction becomes another motivator and staff are less likely to consciously manipulate your measurement structure. There will still be measurement distortion, but it will probably be less pernicious.

Multidimensional: So what should we measure?

- Measuring ongoing effectiveness / performance
  - What are the ***key tasks*** of the employee
    - Writing bug reports?
    - Designing, running, modifying test cases?
    - Developing test strategies / plans / ideas?
    - Editing technical documentation?
    - Writing support materials for the help desk or field support?
    - Facilitate inspections / reviews?
    - Requirements analysis—meet, interview, interpret needs of stakeholders?
    - Release management? Archiving? Configuration management? Buildmeister?
  - Different employees have different key tasks

Multidimensional: So what should we measure?

- Measuring ongoing effectiveness / performance
  - Improvement / education
    - What knowledge and skills did you want the employee to develop?
    - Is it beneficial to your group if the employee studies C++? Management? Philosophy? Theoretical math?
  - Personal attributes as they affect performance
    - Integrity / honesty / trustworthiness / effort to keep commitments
    - Courage
    - Reliability (commitments)

Multidimensional: So what should we measure?

- Differentiate this from snapshots to assess potential performance
  - New hires
  - New manager assesses current staff
  - For this, see my paper on recruiting testers, and evaluation of Knowledge, Skills, Abilities and Other personal attributes.
    - Write me at kaner@kaner.com

Qualitative

- My primary sources of information are not numeric.
  - I don't count bugs or lines of code or hours at work.
  - To gain an impression of her thinking and her work:
    - I review specific artifacts of the tester
    - I discuss specific performances with the tester, preferably while she is performing them (rather than during Evaluation Week)
    - I discuss the work of the tester with others
- Examples:
  - Artifacts: bug reports
  - Performances: test cases and risk analysis
  - Impact: scheduling

Editing Bugs—First impressions

- Is the summary short (about 50-70 characters) and descriptive?
- Can you understand the report?
  - As you read the description, do you understand what the reporter did?
  - Can you envision what the program did in response?
  - Do you understand what the failure was?
- Is it obvious where to start (what state to bring the program to, to replicate the bug)?
- Is it obvious what files to use (if any)? Is it obvious what you would type?
- Is the replication sequence provided as a numbered set of steps, which tell you exactly what to do and, when useful, what you will see?

Editing Bugs—First impressions

- Does the report seem too long? Too short? Does it seem to have a lot of unnecessary steps? (This is your first impression—you might be mistaken. After all, you haven't replicated it yet. But does it LOOK like there's a lot of excess in the report?)
- Does the report seem overly general ("Insert a file and you will see" – what file? What kind of file? Is there an example, like "Insert a file like blah.foo or blah2.fee"?)
- Does the report include unnecessary information, personal opinions or anecdotes that seem out of place?
- Is the tone of the report insulting? Are any words in the report potentially insulting?

Editing Bugs—Replicate the Report

- Can you replicate the bug?
- Did you need additional information or steps?
- Did you get lost or wonder whether you had done a step correctly? Would additional feedback (like, “the program will respond like this...”) have helped?
- Did you have to guess about what to do next?
- Did you have to change your configuration or environment in any way that wasn’t specified in the report?
- Did some steps appear unnecessary? Were they unnecessary?
- Did the description accurately describe the failure?
- Did the summary accurate describe the failure?
- Does the description include non-factual information (such as the tester’s guesses about the underlying fault) and if so, does this information seem credible and useful or not?

Editing Bugs—Follow-Up Tests

- Are there follow-up tests that you would run on this report if you had the time?
  - *In follow-up testing, we vary a test that yielded a less-than-spectacular failure. We vary the operation, data, or environment, asking whether the underlying fault in the code can yield a more serious failure or a failure under a broader range of circumstances.*
  - *You will probably NOT have time to run many follow-up tests yourself. For evaluation, my question is not what the results of these tests were. Rather it is, what follow-up tests should have been run—and then, what tests were run?*
- What would you hope to learn from these tests?
- How important would these tests be?

Editing Bugs—Follow-Up Tests

- Are some tests so obviously probative that you feel a competent reporter would have run them *and described the results?*
    - The report describes a corner case without apparently having checked non-extreme values.
    - Or the report relies on other specific values, with no indication about whether the program just fails on those or on anything in the same class (what is the class?)
    - Or the report is so general that you doubt that it is accurate ("Insert any file at this point" – *really? Any file? Any type of file? Any size? Did the tester supply* reasons for you to believe this generalization is credible? Or examples of files that actually yielded the failure?)

Editing Bugs—Tester's evaluation

- Does the description include non-factual information (such as the tester’s guesses about the underlying fault) and if so, does this information seem credible and useful or not?
- Does the description include statements about why this bug would be important to the customer or to someone else?

The report need not include such information, but if it does, it should be credible, accurate, and useful.

Qualitative measurement: Performances

- Weekly personal review
  - In the tester's cube, not mine
  - Show me your best work from last week
  - Show me your most interesting bugs
  - Show me your most interesting test cases
  - What have you been testing? Why did you do it that way? Have you thought about this?
- Risk analysis
  - Review an area (e.g. functional area) of the program under test by this tester
    - What are the key risks?
    - How do you know?

Qualitative measurement: Performances

- Risk analysis
  - Review an area (e.g. functional area) of the program under test by this tester
    - What are the key risks?
    - How do you know?
    - How are you testing against these risks?
    - How are you optimizing your testing against these risks?

Qualitative measurement: Performance

- Scheduling
  - Weekly status: Accomplishments and Objectives
    - Brief
    - Always list last week's objectives and this week's accomplishments against them
    - Show accomplishments that meet larger objectives but were not anticipated for last week (serendipity is OK, so is shifting gear to protect productivity)
  - Task lists
    - Projected time planned per task
    - Estimated time spent per task
    - Estimated work (% of task) remaining to do

Qualitative measurement: Impact

- The essence of impact measurement is its effect on others. Generally, you collect these data from other people.
  - 360-Reviews often suffer from lack of specificity, that is, lack of impact measurement. Here are a *few illustrations of more specific, open-ended, behavioral questions*
    - What tasks were you expecting or hoping for from this person
    - What did they do
    - Please give examples
    - How pleased were you
    - What could they improve
    - What was left, after they did their thing, that you or someone else still needed to do
    - How predictable was their work (time, coverage, quality, communication)

Multi-sourced (e.g. The 360 review)

- You are only one perceiver
- The tester provides services to others
  - Evaluate the tester's performance by
    - Examining the tester's work products for others
    - Interviewing the others, typically supported by a standard questionnaire
  - Make sure you interview people with different interests
    - Tech support and documentation, not just programmers and project managers
  - Ask performance oriented questions, not just how “good' the tester is
  - Look for patterns across people / groups. If there is no consistency, why not?

Based on multiple samples

- Not just one project
- Not just bug reports for one programmer
- Not just one test plan
- Not just one style of testing
- Not just performance this month

- It's not enough to be fair (with evaluation spread over time). It is essential that you be perceived to be fair. Lack of perceived fairness drives your relationship with the employee back to the 2-party or 3-party model baseline.

Individually tailored

- Different people do different tasks
  - A toolsmith shouldn't be evaluated on the quality of her bug reports
  - An exploratory tester shouldn't be evaluated on the quality of his test documentation
  - A test lead should be evaluated on the performance of her staff

Feedback or layoff, but not for quantifying the raise

- Raises
  - Often infected with popularity issues (especially when there is “levelling” and executive participation)
  - Are typically cost of living or less
  - Are often only weakly tied to actual performance or value to the company
  - Are often out of your control
- Minimize the demotivational impact of raises by separating them from the performance evaluation
  - Best use of performance evals (frequent) is coaching
  - A sometimes essential use is to establish your bargaining position with respect to layoffs.