# Metrics, Qualitative Measurement, and Stakeholder Value

Conference of the Association for Software Testing
July 2009


## Cem Kaner, J.D., Ph.D.

Professor of Software Engineering
Florida Institute of Technology



These notes are partially based on research that was supported by NSF Grants EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers" and CCLI-0717613 "Adaptation & Implementation of an Activity-Based Online or Hybrid Course in Software Testing." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

# Abstract

The same old platitudes about measurement don't help us. Most companies don't have measurement programs. That's not because seasoned software executives have no experience with measurement programs. It's that they have too much. It's not because they're lazy. It's because we do software-related measurement so badly that the measurement programs do more harm than good. We need measurements that provide genuine value, that are worth their costs (direct and indirect).

Genuinely valuable measurements help answer questions that stakeholders consider important.

This tutorial has four parts:

- Facilitated discussion that yields a list of important questions.
- Brief review of some traditional measurements offered for these questions.
- Review of traditional measurement theory, with an emphasis on validity and threats to validity, specifically applied to the some of the measures on our list.
- Overview of qualitative measurement, with application to some of the questions on our list. We'll close with an effort to pull this together into some recommendations for application on the job.

Qualitative measurements don't give you numbers. Instead, they provide well-organized descriptions that reflect the complexity and detail of the underlying tasks, behaviors or problems. They can provide a powerful foundation for comparisons, evaluations, or examples that illustrate or challenge ideas from other sources. In many ways, the contrast between qualitative and traditional measurement is like the contrast between scenario testing and automated build verification. Both have their uses and their limits. Neither tells the full story. A blending of methods across the spectrum (what we call "mixed methods" in measurement theory) provide a richer set of descriptions, and a better basis for making decisions.

# Note

Some people read slide sets from my website and misunderstand what was taught.

If you are looking for a basic discussion of metrics (from my viewpoint), look at www.kaner.com/articles.html for

- Cem Kaner, "Software-related measurement: Risks and opportunities"
- Cem Kaner & Stephen J. Swenson, "Good enough V&V for simulations: Some possibly helpful thoughts from the law & ethics of commercial software." Simulation Interoperability Workshop, Providence, RI, April 2008
- Cem Kaner & Walter P. Bond, "Software engineering metrics: What do they measure and how do we know?" *10th International Software Metrics Symposium (Metrics 2004)*, Chicago, IL, September 14-16, 2004.
- Cem Kaner, Elisabeth Hendrickson & Jennifer Smith-Brock , "Managing the proportion of testers to (other) developers." *Pacific Northwest Software Quality Conference*, Portland, OR, October, 2001.

Today's slides are for a tightly time-limited, interactive tutorial. My understanding is that the people who signed up for it have been to talks of mine on metrics in the past and have a pretty clear idea of many of the basics that these slides will not develop thoroughly.

If you know the material, these slides might give you thoughts on how to teach it. If you think there are seriously controversial things here to criticize, please look at one of the other sources for a more detailed explanation rather than a fast sketch that is intended to draw in-room discussion.

1. Facilitated discussion that yields a list of important questions.

What do you want to measure?

(Question)

Why?

(Goal)

## A few of my questions (and goals), in case we need them

- How complex is this software?
- How good is this tester?
- What are our projected support costs?
- What is the reliability of this software?
- How can we improve the maintainability of this code?
- Who should I fire?
- Can we ship it yet?
- How close to done are we?

# GQM

- Goal
  - (what are you trying to learn? What decision are you trying tot make?)
- Question
  - what information is most important for achieving your goal?
- Metric
  - what measurements will yield the information that will help you achieve your goal?
- https://www.goldpractices.com/practices/gqm/ is a reasonably good summary, that points out some of the important complexities

2. Brief review of some common measurements offered for these questions.

What do you want to measure?

(Question)

How should we measure it?

(Metric)

| What do you want to measure? | How should we measure it? |
| --- | --- |
| • How complex is this software? | • McCabe's psychosomatic complexity metric?<br>• Structural complexity?<br>• Semantic complexity?<br>• Bug counts? |
| • How good is this tester? | • Bug counts?<br>• Average bug severity?<br>• How many bugs get fixed?<br>• Rate of creation of (automated?) tests? |
| • What are our projected support costs? | • Historical call volumes for comparable products?<br>• Help desk staff rating?<br>• Bug counts? |
| • What is the reliability of this software? | • Bug counts (the duct tape of modern QA)?<br>• Percent mutations found?<br>• Percent seeded bugs found?<br>• Mean time between failures in testing? |

3. Review of traditional measurement theory, with emphasis on validity and threats to validity, specifically applied to some measures on our list.

# What makes these measurements good?

- (This is my own idiosyncratic list. You can find others with any search engine...)
- Relevance / Utility
  - Do you learn something that you want to learn (e.g. decision support)?
- Precision
  - Consider a set of measurements, $\mathcal{M}$, whose members differ from each other only in precision of the calculations used to obtain them.
  - If M is the exact underlying "true" value, then we could argue that any m $\in \mathcal{M}$ is precise enough if it would lead to the same decision as M.
  - From the standpoint of precision, an estimate might be very rough but be precise enough.

# What makes these measurements good?

- Credibility
  - When the VP in charge of not believing you challenges your numbers and your conclusion,
    - the metric that he believes is most accurate is probably your most credible (with him)
    - a low-precision estimator may lack credibility even if it is operationally accurate
- Repeatability
  - If taking the same measurements won't yield the same results (plus or minus a relatively small allowance for measurement error), no one will (or should) trust your results. Next time you look, they will be different.

# What makes these measurements good?

- Practicality:
  - How difficult / expensive is it to actually take this measurement?
- Proneness to side effects:
  - What are the implications of taking and using this measurement? How will this change the system being measured? What are the unintended but likely consequences?
- Abusability :
  - What evil can people accomplish when armed with these measurements? Within "evil", I include the road to hell paved with good intentions
- Potential benefits:
  - What good can people accomplish when armed with these measurements?

# What makes these measurements good?

- Privacy:
  - Are things that should be kept private actually kept private? Or, now that a measurement has become practical, have historically accepted privacy interests become mere barriers to perceived progress?
- Appropriateness of scope and use:
  - Consider the same measurement (e.g. rate of errors of a certain kind):
    - taken by a programmer of her own work, or
    - by a programmer in conjunction with a coach, or
    - by a development group that collectively improves their work but keeps the data private from outsiders, or by a
    - technical team that will report the data to management external to the group.

# What makes these measurements good?

- Validity:
  - Are you measuring what you think you are measuring?
  - People sometimes operationalize this by asking whether you can reasonably expect to get the same result if the variable you think you are measuring stays the same but many other conditions are different?

# What makes these measurements good?

- In my experience in computing,
  - the ethical questions are rarely discussed in a constructive, practical way (other than the practicality of dismissing them)
  - and the validity of measurements is usually
    - ignored, or
    - treated as obvious and minor
    - dealt with in the context of formalized mathematical models that no one expects to really apply fully anyway

As an easily accessible example from a prestigious source, look at Robert E. Park, Wolfhart B. Goethert, William A. Florac (1996) *Goal-Driven Software Measurement—A Guidebook*, Software Engineering Institute, http://www.sei.cmu.edu/pub/documents/96.reports/pdf/hb002.96.pdf

# What makes these measurements good?

- Achieving validity is probably
  - the most difficult task in measurement of most attributes
  - the most important task
  - the root of the most severe negative consequences if we don't achieve it
- In other disciplines (e.g. throughout the social sciences), validity is a core area of discussion in measurement, often the most thoroughly discussed (especially in undergraduate training, which looks more at the broad issues in the field and less at the mechanics)
- Because validity is so lightly and loosely (and rarely) considered in computing (undergrad, grad, or as applied), I think most traditional SE metrics are
  - worthless (at best) and
  - more likely to do serious harm than good (see Robert Austin, Measuring and Managing Performance in Organizations)

# External validity (From Wikipedia, the free encyclopedia)

**External validity** is the validity of generalized (causal) inferences in scientific studies, usually based on experiments as experimental validity.

Inferences about cause-effect relationships based on a specific scientific study are said to possess external validity if they may be generalized from the unique and idiosyncratic settings, procedures and participants to other populations and conditions Causal inferences said to possess high degrees of external validity can reasonably be expected to apply (a) to the target population of the study (i.e. from which the sample was drawn) (also referred to as population validity), and (b) to the universe of other populations (e.g. across time and space).

The most common loss of external validity comes from the fact that experiments using human participants often employ small samples obtained from a single geographic location or with idiosyncratic features (e.g. volunteers). Because of this, one can not be sure that the conclusions drawn about cause-effect-relationships do actually apply to people in other geographic locations or without these features.

# Threats to external validity (From Wikipedia, the free encyclopedia)

"A threat to external validity is an explanation of how you might be wrong in making a generalization." Generally, generalizability is limited when the cause (i.e. the independent variable) depends on other factors; therefore, all threats to external validity interact with the independent variable.

**Aptitude-Treatment-Interaction:** The sample may have certain features that may interact with the independent variable, limiting generalizability. For example, inferences based on comparative psychotherapy studies often employ specific samples (e.g. volunteers, highly depressed, no comorbidity). If psychotherapy is found effective for these sample patients, will it also be effective for non-volunteers or the mildly depressed or patients with concurrent other disorders?

**Situation**: All situational specifics (e.g. treatment conditions, time, location, lighting, noise, treatment administration, investigator, timing, scope and extent of measurement, etc. etc.) of a study potentially limit generalizability.

**Pre-Test Effects:** If cause-effect relationships can only be found when pre-tests are carried out, then this also limits the generality of the findings.

**Post-Test Effects**: If cause-effect relationships can only be found when post-tests are carried out, then this also limits the generality of the findings.

**Reactivity** (Placebo, Novelty, and Hawthorne Effects): If cause-effect relationships are found they might not be generalizable to other settings or situations if the effects found only occurred as an effect of studying the situation.

**Rosenthal Effects:** Inferences about cause-consequence relationships may not be generalizable to other investigators or researchers.


Construct validity refers to the degree to which inferences can legitimately be made from the operationalizations in your study to the theoretical constructs on which those operationalizations were based. Like external validity, construct validity is related to generalizing. But, where external validity involves generalizing from your study context to other people, places or times, construct validity involves generalizing from your program or measures to the *concept* of your program or measures. You might think of construct validity as a "labeling" issue. When you implement a program that you call a "Head Start" program, is your label an accurate one? When you measure what you term "self esteem" is that what you were really measuring?



http://www.socialresearchmethods.net/kb/constval.php

A good academic discussion of threats to construct validity is at
http://www.indiana.edu/~educy520/sec5982/week_3/construct_validity_trochim.pdf

# Some threats to construct validity (associated with metrics)

1. No explicit construct (or as Trochim puts it, "Inadequate Preoperational Explication of Constructs")
   - **Example: what is code complexity?**
2. The measured values might be influenced by the value of the construct AND by other values, thus changing the measured value might be accomplished by changing a non-construct value
   - **Measured value = number of branches. Does greater complexity change the number of branches? Can we change # of branches without a change in complexity?**
3. The measured value might influence the value of the construct, but be only one of the factors that changes the construct
   - **Is number of branches the primary determiner of complexity? Can we change complexity w/o changing # branches?**
4. Measured values might be correlated with the construct but not causally connected
5. No theory mapping the metric values to the construct

# Some threats to construct validity (associated with metrics)

"In a research study you are likely to reach a conclusion that

- your [research] program was a good operationalization of what you wanted and that
- your measures reflected what you wanted them to reflect.

Would you be correct? How will you be criticized if you make these types of claims? How might you strengthen your claims.

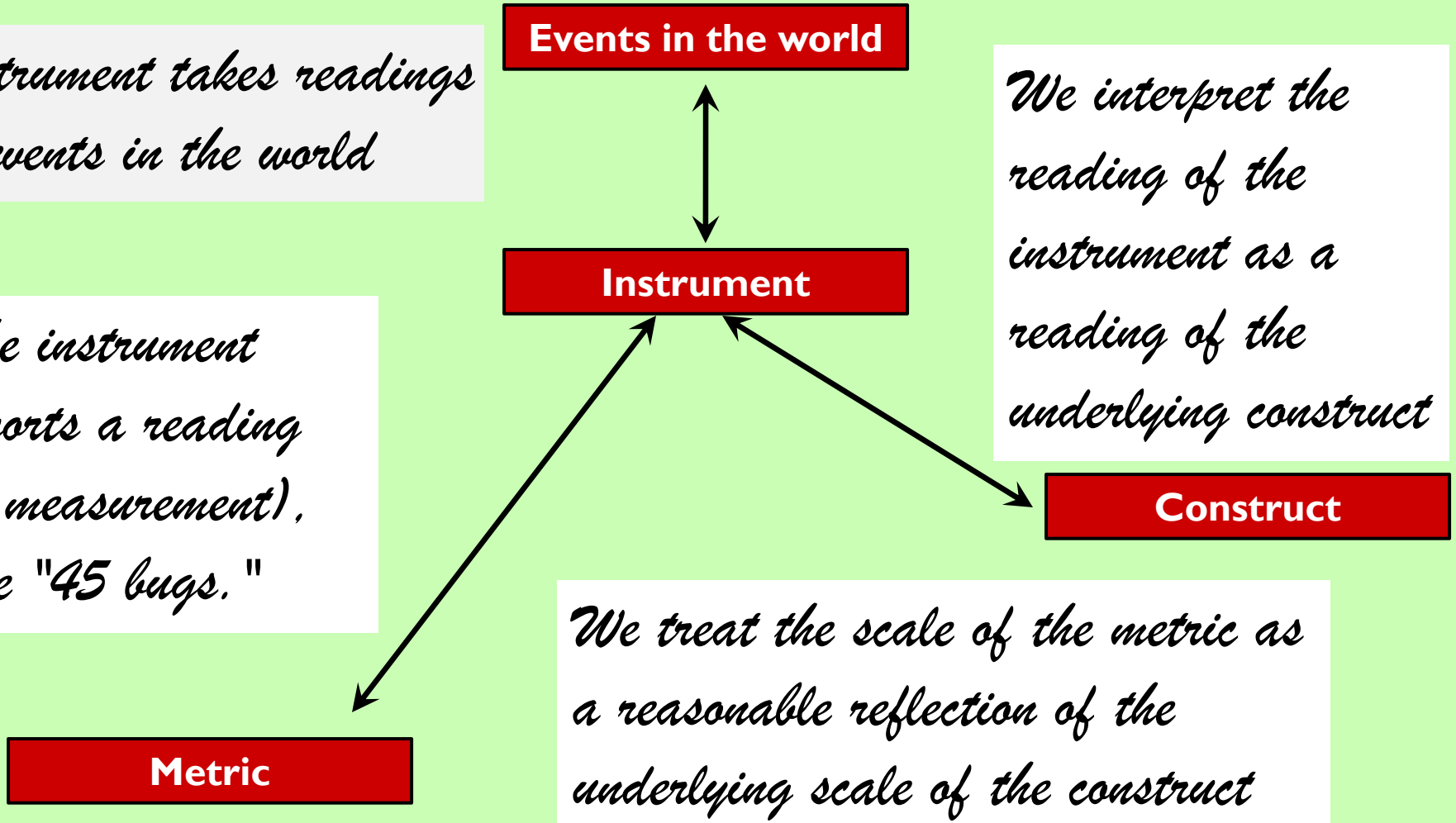
The kinds of questions and issues your critics will raise are what I mean by threats to construct validity."

http://www.indiana.edu/~educy520/sec5982/week_3/construct_validity_trochim.pdf

# Constructs and Metrics

*Research program studies the instrument as a measuring device for the construct*

*Instrument takes readings of events in the world*




*We interpret the reading of the instrument as a reading of the underlying construct*

*The instrument reports a reading (a measurement), like "45 bugs."*

*We treat the scale of the metric as a reasonable reflection of the underlying scale of the construct*

| What do you want to measure? | How should we measure it? |
| --- | --- |
| • How complex is this software? | • McCabe's psychosomatic complexity metric?<br>• Structural complexity?<br>• Semantic complexity?<br>• Bug counts? |
| • How good is this tester? | • Bug counts?<br>• Average bug severity?<br>• How many bugs get fixed?<br>• Rate of creation of (automated?) tests? |
| • What are our projected support costs? | • Historical call volumes for comparable products?<br>• Help desk staff rating?<br>• Bug counts? |
| • What is the reliability of this software? | • Bug counts (the duct tape of modern QA)?<br>• Percent mutations found?<br>• Percent seeded bugs found?<br>• Mean time between failures in testing? |

# A more explicit analysis

| Construct | Instrument | Metric | Validity |
| --- | --- | --- | --- |
| | | | |

4. Overview of qualitative measurement, with application to some of the questions on our list.

We'll close with an effort to pull this together into some recommendations for application on the job.

# Consider the question: How good is this tester?

Let me start with the obvious

# What does this tester actually DO?

*Testing: An empirical, technical investigation of the product under test conducted to provide stakeholders with quality-related information.*

Testers (in general) (what about THIS ONE?):

- Design tests
- Implement tests
- Execute tests
- Report bugs
- Document tests and test results

These carry a lot of complexity. For example:

- **Design tests includes** (just to name a few)
  - what techniques does this person know and use?
  - how suited are these techniques for the mission and context?
  - how well does he apply them?
  - what is the theory of coverage?
  - what is the theory of reuse and maintainability?

# What does this tester actually DO?

Different testing missions (different information objectives) should lead to different testing. What is the mission assigned to this tester?

- Find important bugs, to get them fixed
- Assess the quality of the product
- Help managers make release decisions
- Block premature product releases
- Help predict and control costs of product support
- Check interoperability with other products
- Find safe scenarios for use of the product
- Assess conformance to specifications
- Certify the product meets a particular standard
- Ensure the testing process meets accountability standards
- Minimize the risk of safety-related lawsuits
- Help clients improve product quality & testability
- Help clients improve their processes
- Evaluate the product for a third party

# But WAIT, there's MORE! What about tasks like these? Does this tester do them? Is he supposed to?

- Write requirements
- Participate in inspections and walkthroughs
- Compile the software
- Write installers
- Investigate bugs, analyzing the source code to discover the underlying errors
- Conduct glass box tests
- Configure / maintain programming-related tools, e.g. source control system
- Archive the software
- Evaluate reliability of components the company is thinking of using
- Provide technical support
- Train new users (or tech support or training staff) in the use of the product
- Demonstrate the product at trade shows or internal company meetings
- Provide risk assessments
- Collect and report statistical data (software metrics) about the project
- Build and maintain internal test-related tools such as the bug tracking system
- Benchmark competing products
- Evaluate marketplace significance of hardware/software configurations (to inform choice of configuration tests)
- Conduct usability tests
- Lead / audit efforts to comply with regulatory / industry standards (such as those published by SEI, ISO, IEEE, FDA)
- Provide a wide range of project management services.

# What does this tester actually DO?

Oh, by the way:

- What does THIS TESTER think he is supposed to do?
- And why does he think that?
- And how do you know?
- And if his impression is different from your impression
  - should you assess his work against his impression
  - or against yours?

# Too Much Data?

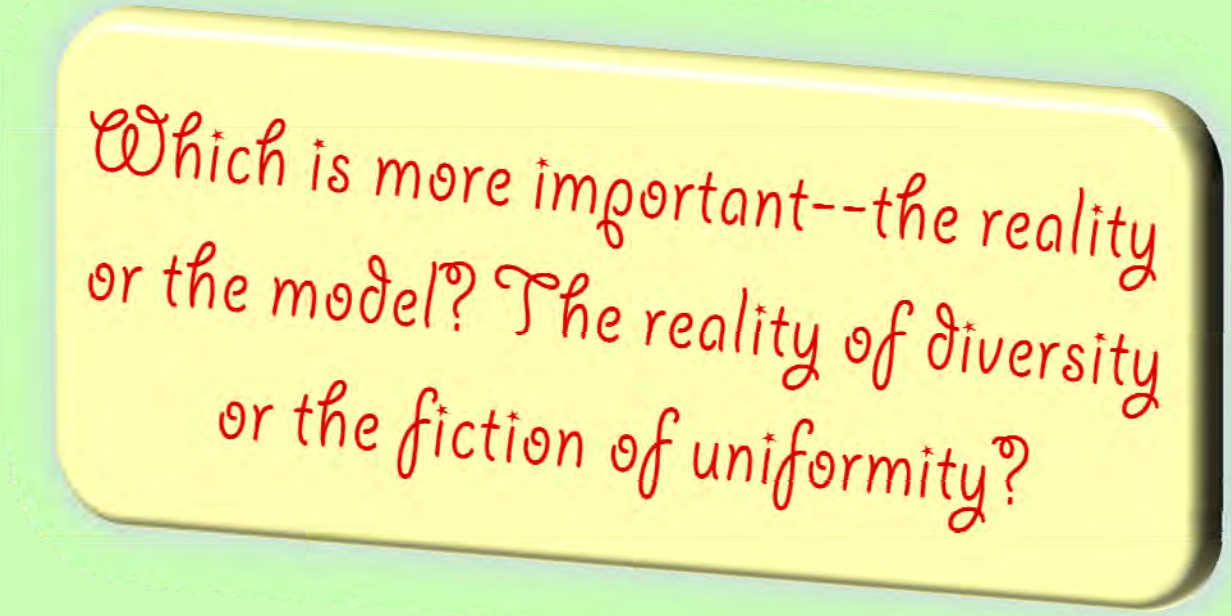
- Yes, we risk analysis/paralysis
- Yes, we risk doing nothing because the cost of doing everything is impossibly high
- Yes, the perfect is the enemy of the good enough
- But really,
  - Before we start nailing our numbers on this guy, can't we at least TRY to figure out
    - What are the top 5 ways this person spends his time?
    - What are the 5 most important tasks (in his view)?
    - What are the 5 most important tasks (in our view)?
    - What are the 5-10 tasks we want to assess him against?

# Uh Oh!

We're not going to evaluate everyone against the same criteria!

Maybe people who do different things should be evaluated differently?

Maybe it's OK (or even if it's not OK, maybe it's true) that people do different things?



# How do you find this out?

Meet with the tester

- simple direct questions, but they can take time and trust
  - what does a typical day look like?
  - what are examples of your best recent work?
  - what are examples of your most important recent work?
  - what things should you really be doing but you feel blocked or undertrained or in some other way unable to do as well as you should?
  - what things do you really want to start doing?
  - what are the most important tasks other people in the group are doing?
  - what should you stop doing?
  - what should our group stop doing?

# How do you find this out?

Gather samples of the tester's work products

- What types of work are there?
- Are there other work products that you aren't seeing (ask the tester)?
- Are you competent to review these? (If not, don't. Either get help or *risk alert* skip them)
- Is this a representative sample? (It doesn't have to be representative as long as it is fair / useful) but you need to know about representativeness if you plan any kind of quantitative summary

# Gathering Information from Others

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 in how people evaluate this person, why not?

# Base Assessment on Multiple Samples

Measurement error arises in qualitative analysis, at least as much as statistical process control

- 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 to an adversary model that will not lead to sustained improvements.

# Start evaluating the work

Review the tester's work products

- Skim to separate
  - Looks good
  - Maybe OK
  - Horrible

# You MAY have a task analysis or task assessment guide

Here are two examples:

- Evaluating bug reports
- Evaluating risk analysis

# Evaluating Bug Reporting

Imagine taking a sample of an employee's bug reports, perhaps 10 reports filed over a 6-month period.

How can you decide how good each report is?

Here are some training guidelines that I developed (based on what colleagues and I have done) for appraising the quality of a bug report.

This list is not a checklist. You don't want to answer every question—that would take too long.

But as you review the report and the bug it describes, these suggestions will give you ideas of questions you might want to ask next.

# Start by Forming a First Impression

***Don't try to replicate the bug yet. Just skim the report and think about your reaction. What have you learned form the report, and how hard was it to learn it?***

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?

# First Impression

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?

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?

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"?)

# Replicate the Report

***Now, replicate the bug. How well was it actually described?***

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?

# Replicate the Report

Did the description accurately describe the failure?

Did the summary accurate describe the failure?

What about statements that convey the tester's judgment?

- 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.***

# 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?

# Follow-Up Tests

Are some tests so obviously probative that you feel a competent reporter **would definitely have run them** and described the results? For example,

- A report describes a corner case without apparently having checked non-extreme values.
- A report relies on specific values, with no indication about whether the program just fails on those or on anything in the same class (what is the class?)
- A 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?)

# Another Example: Risk Analysis

- Review an area (e.g. functional area) of the program under test by this tester.
- Here are some questions that I might ask an employee to answer.
  - What are the key risks?
  - How do you know?
  - What is your strategy for identifying and appraising risks?
  - How are you testing against these risks?
  - How are you optimizing your testing against these risks?

# Risk Analysis

Giri Vijayaraghavan wrote a M.Sc. thesis on risk-based testing and then did follow-up research at T.I.

- See his papers at http://www.testingeducation.org/articles

James Bach created the Heuristic Test Strategy Model for walking through the product, quality expectations for the product, and project management factors to identify potential areas of weakness.

- See his model at http://www.satisfice.com/tools/satisfice-tsm-4p.pdf

Both of these are examples of the broader category of techniques called Failure Mode & Effects Analysis.

# Risk Analysis

There are other ways to imagine:

- how the program could fail or
- what tests are interesting from the point of view of the question, what could fail

For more examples, see my course slides and videos at

- www.testingeducation.org/BBST/BBSTRisk-BasedTesting.html

Ultimately, you are trying to figure out:

- what techniques **this tester** uses to identify risks and design tests related to them

Here are some of the questions you might ask in trying to appraise whether the tester is doing this well.

# Risk Analysis: Identifying Risks

- Can the tester articulate her strategy for identifying risks?
- Based on the model you use for identifying risks, can you think of any risk areas that were missed?
- Does the tester think in multidimensional terms (this might fail under these conditions, or these aspects of functionality might conflict when used together)?
- Are the risks appropriate to the business and technical domains?
- Is the tester paying attention to the classes of risk that would count to the stakeholder or distributing her analytical effort arbitrarily?
- How far out of the box does the tester think?

# Risk Analysis: Identifying Risks

- Is the tester consulting sources of failure (e.g. help desk records) to check whether there are classes of potential failures that the risk list doesn't imagine?
- Are other available sources consulted?
- Is the tester getting effective peer review of her lists?
- Given feedback on her risk list (and estimated significance of problems), does the tester actually use that feedback and change the risk list or risks database appropriately?
- Does the tester recognize when she doesn't know how to imagine the details of some classes of risk and get help?
- Does the tester imagine risks associated with interaction with other products / platforms or is she stuck thinking inside the box that contains only this product?

# Risk Analysis: Significance and Credibility

- Can / does the tester develop tests to gain insight into the likelihood of a particular failure or potential cost of it? Can he provide examples of these types of tests?
- Are any of the claimed risks implausible?
- Are the risks communicated effectively? For example, are they tied to business or stakeholder value?
- Would a stakeholder consider this risk analysis sufficiently relevant and comprehensive?
- How does the tester characterize or compute the magnitude of risk? Does he consider both the likelihood of failure in the field (where does he get relevant data?) and potential consequences?

# Risk Analysis: Test Design

Given a list of potential problems in the product:

- Are suitable tests designed for each problem (or each one of sufficient priority)?
  - A suitable test is one that is well suited to exposing the problem if it is present in this product
- Is the test documented in a way that makes it traceable to the risk? (Assuming, which may be incorrect, that you want such documentation.)
- If tests are developed on the basis of risks, is the tester prioritizing test development wisely (highest priority to the worst risks)?
- Does the tester recognize when she doesn't know how to test for a given risk and get help?
- What is the tester's stopping rule for this work?

# What is qualitative analysis?

- Rather than testing quantifiable or Yes/No hypotheses, we are looking for information on
  - what is relevant
  - what is important
  - whether there are key divergences (e.g. different goals for different people) or other important potential sources / effects of variation
  - what "good" means under the circumstances
  - unexpected side-issues that turn out to be relevant
  - emerging themes / impressions (ideas for answers to the above will evolve over time)
- We can do our observations through
  - interviews, artifact analysis (often called "content analysis"), case studies, focus groups, observation of tasks/situations

# What is qualitative analysis?

- Data collection
  - Raw data, of varying types and depth
  - We don't have replications with readings of the same things every time. We have observations of similar things or situations, but they each present their own information
  - Bias is
    - inevitable
    - a serious risk
  - Filtering is
    - inevitable
    - a problem when it is due to bias
  - Interpretation and selectivity are:
    - essential, unavoidable, subject to bias

# What is qualitative analysis?

- It is extremely useful to work in several cycles:
  - Make observations (collect data)
  - Analyze data
    - generate some questions
    - realize that some data you were ignoring is relevant after all and should be collected next time
  - More data collection
  - More analysis
    - reanalyze old data as you gain more understanding and new questions
  - Vary the data collection methods
  - Vary the types of artifacts examined
  - Vary the people / roles used as sources
  - (We are looking at what we are studying from may perspectives. Qualitative analysts discuss this under the term, "triangulation")

# For discussion: Objections to qualitative analysis

- "These are just anecdotes" MEANS
  - "I don't believe your results or conclusions will transfer" (will correctly describe other reasonably similar situations) (I don't think your work has external validity)
  - "I think your observations are biased, that what you are telling me has more to do with your opinion than with what you saw"
  - "I don't like what you are saying and so I can dismiss it as subjective"
- "We don't have a basis for comparison"
  - for example, how do you rank one tester over another?
- "We don't have clear summaries"
- "How can I communicate my results to management?"

# More on dealing with objections

The objections point to genuine risks. Here are some things to keep in mind:

1. The researcher IS the measuring instrument (see next slide), so we have to work with that.
2. In your cycles of observation and analysis, it is essential to look for disconfirmations as well as confirmations. The data are noisy, so there should always be counterexamples. Pay attention to them.
3. Much of your presentation should be in the words of the people who spoke to you, or should show the critically relevant parts of the artifacts you reviewed. This is a detailed summary of the data, that will be the background for the short summaries you might prepare to management. We discuss this more in our talk on CHAT.
4. Management summaries are inevitable. Executives who want an answer that they can process in 1-5 minutes will not be willing to read a compendium. However, once they (or their staff) understand your summary, they may want to discuss it. That's when your report is critical.

# Researcher as Instrument (Strauss & Corbin, 1998; Punch, 1998)

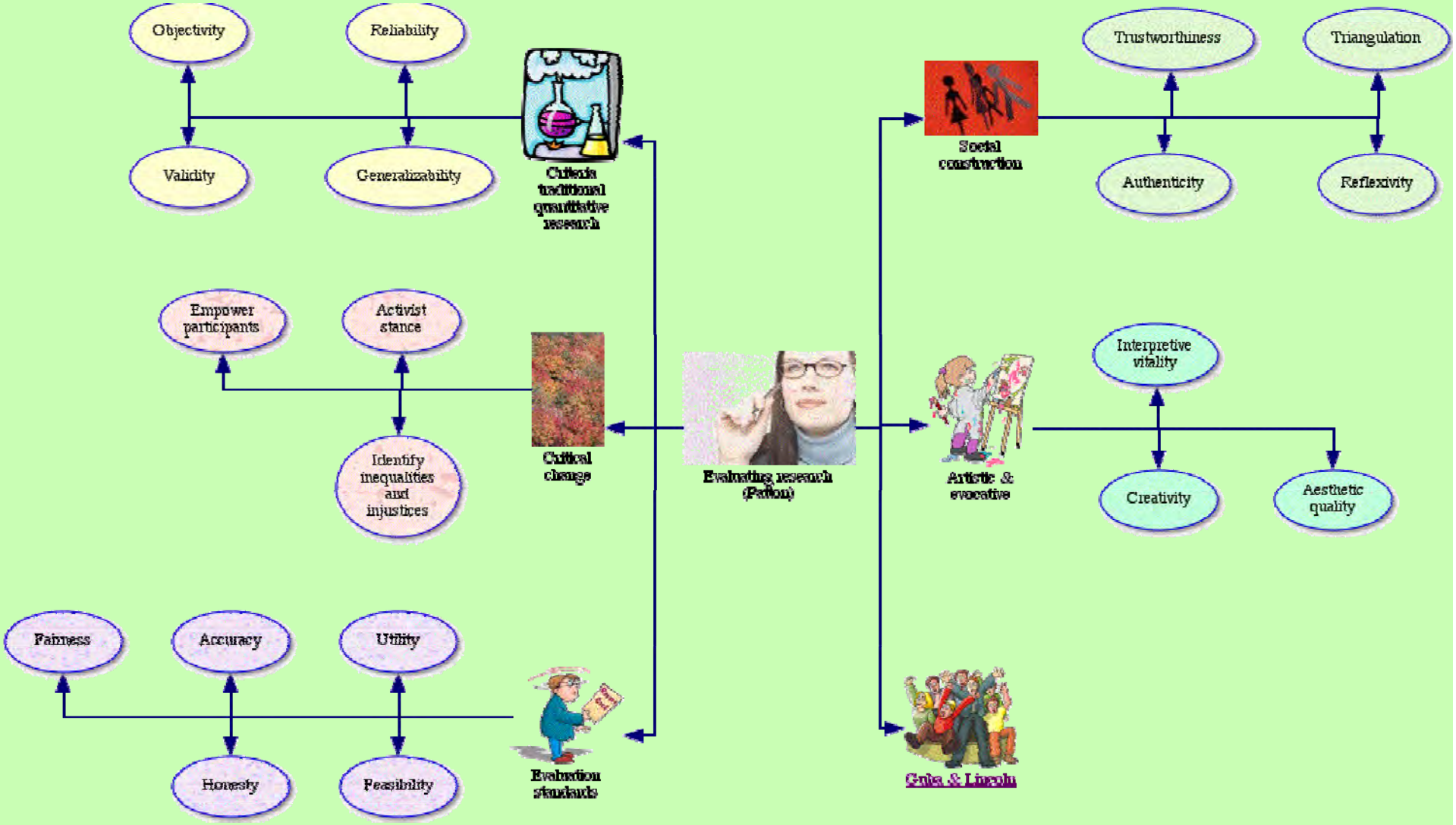
Rather than use systematically developed, carefully vetted instruments, the qualitative researcher:

- Uses her own observations and perceptions
- Positions herself in a unique and rapidly unfolding field situation
- Collects, analyzes, and reports data through her eyes

Data are interpreted through the researcher's worldview (Rossman & Rollis, 2003)

# Coping with researcher bias (reflexivity)

- Reflect on your role and purpose before beginning the work
  - Understand your own beliefs, values, assumptions, and biases. Be on guard against them. Rossman & Rallis (2003)
  - Clarify theoretical and methodological orientation ahead of time
- Be constantly aware of the role, interactions, and materials throughout the project (Delamont, 2002)
- Use critical friends, member checks, triangulation, and prolonged engagement to enhance credibility




Slide from Rebecca Fiedler, ideas from Patton (2001)

From Rebecca Fiedler

Criteria traditional quantitative research

Guba & Lincoln criteria for qualitative research

- Validity
- Generalizability
- Objectivity
- Reliability
- Credibility
  - Reflexivity
  - Prolonged engagement
  - Triangulation
    - Across methods
    - Across informants
  - Critical friends
  - Member checks
- Transferability
  - Offer descriptions
- Dependability
  - Rigor of the process
    - Discussion of methods
    - Disclosed thinking behind choices
- Confirmability
  - Evidence to support claims and interpretations
    - Liberal use of participant quotes
    - Confirming & disconfirming evidence

# One more example

What SHOULD we mean by code complexity?

How SHOULD we measure code complexity?

How can we figure this out?

# References


- Austin, R.D. (1996) *Measuring & Managing Performance in Organizations*, Dorset House Publishing
- Creswell, J.W. (2nd. ed. 2006) *Qualitative Inquiry & Research Design: Choosing among Five Approaches*, Sage Publications.
- Creswell, J.W. (3rd ed. 2008) *Qualitative, Quantitative, & Mixed Methods Approaches*, Sage Publications.
- Delamont, S. (2002). *Fieldwork in educational settings: Methods, pitfalls and perspectives*. Routledge.
- Denzin, N.K., & Lincoln, Y. (eds) (2005) *The SAGE Handbook of Qualitative Research*, Sage Publications.
- Guba, E. G., & Lincoln, Y. S. (1989). *Fourth generation evaluation*. Sage Publications.
- Lincoln, Y. & Guba, E. (1985) *Naturalistic Inquiry*, Sage Publications.
- Patton, M.Q. (2001) *Qualitative Research & Evaluation Methods*, Sage Publications.
- Punch, M. (1998). Politics and ethics in qualitative research. In N. K. Denzin & Y. S. Lincoln (Eds.), *The landscape of qualitative research*: Theories and issues (pp. 470). Sage Publications.
- Rossman, G. B., & Rallis, S. F. (2003). *Learning in the field: An introduction to qualitative research* (2nd ed.). Sage Publications.
- Strauss, A., & Corbin, J. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (2nd ed.). Sage Publications.
- Trochim, W. (undated) *Web Center for Social Research Methods*, http://www.socialresearchmethods.net/