

Metrics and Measures and Numbers, Oh My!

Michael Bolton, DevelopSense
Toronto, Ontario, Canada
mb@developsense.com
<http://www.developsense.com>
+1 (416) 656-5160

Bucharest
December 2014

Notes & Acknowledgements

- Acknowledgements to my colleagues and mentors:
 - James Bach
 - Jon Bach
 - Cem Kaner
 - Jerry Weinberg
- Get the complete version of this presentation
 - from my USB key
 - with a business card
 - from <http://www.developsense.com/past.html>

These notes are rough drafts. I won't talk about everything in them, and I will talk about stuff NOT in them.

Some people have strange ideas about
software development.

They think of THIS as "efficiency"...



image credit: istockphoto.com

But they forget that
that the product was DESIGNED.

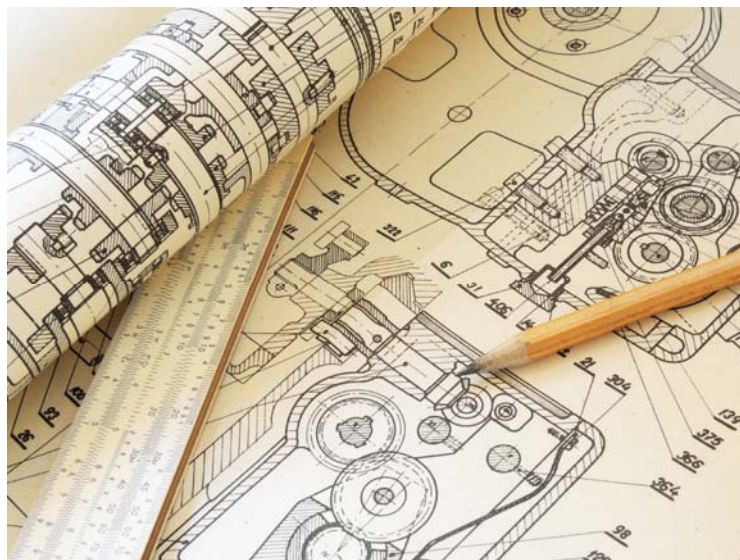


image credit: istockphoto.com

And they forget that
the FACTORY had to be built.



image credit: istockphoto.com

AND they forget that
the factory had to be DESIGNED.



image credit: istockphoto.com

Software development is not factory work.
It's not about the part where we make
a million copies of the same thing.

Software development is the part where we
design the thing that we're going to copy.

Plus: notice anything missing from all this?

They forget that PEOPLE did all this stuff.

The people with the weird ideas
forget that designs must be DEVELOPED.



image credit: istockphoto.com

They forget that designs
must be INTERPRETED.



image credit: istockphoto.com

They forget that putting even well-designed things together is often MESSY.



image credit: istockphoto.com

They forget that even building MODELS can be messy.



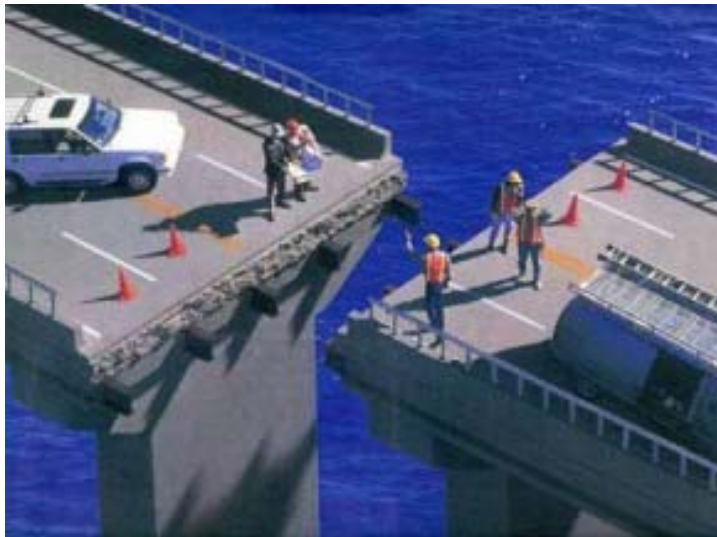
image credit: istockphoto.com

All they seem to know about building models is what they can see of the finished product.



image credit: istockphoto.com

They forget what happens when you try to complete too much work too fast.



They forget what happens
when you try to drive too *quickly*
without driving *carefully*.



We develop skill and tacit knowledge in three ways.
Only one way is through explanations,
and that may be the weakest way.

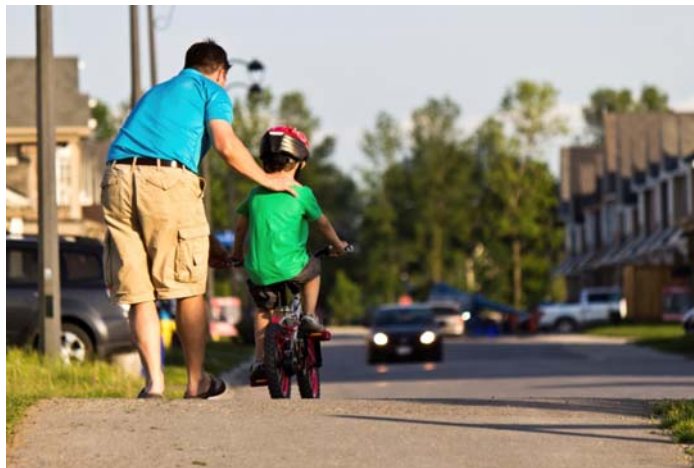


image credit: istockphoto.com

We also learn from being immersed in a culture where people are doing the same kinds of things. Many process models fail to recognize this.



image credit: istockphoto.com

We learn far more, and more quickly, from our own practice, mistakes, and feedback. Many process models fail to recognize feedback loops.



image credit: istockphoto.com

People forget that even though we would prefer not to make mistakes, mistakes are normal when we're learning to do difficult things.



image credit: istockphoto.com

They forget that problems happen even when we're NOT trying to learn difficult things.



image credit: istockphoto.com

They become fixated on numbers and charts and dashboards.



They forget that when you're driving, it's important to look out the window too.



image credit: istockphoto.com

They stop believing that you can recognize significant, important things without numbers.



They forget that numbers are *illustrations*: extra data, not information.

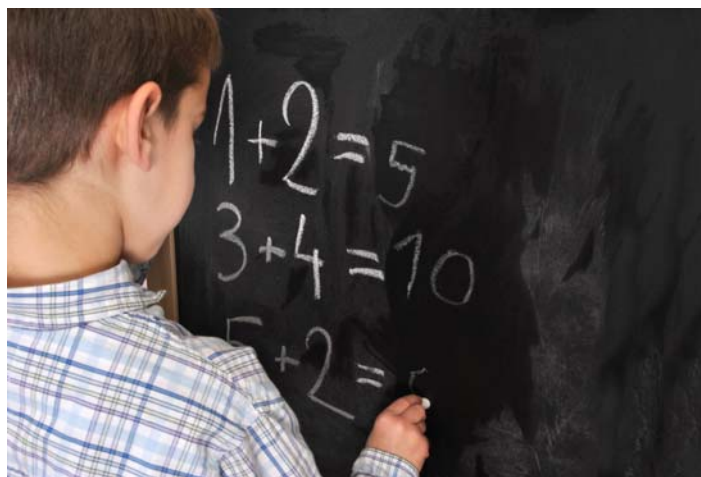


image credit: istockphoto.com

They decide that test cases are a good idea,
because you can COUNT test cases.



image credit: istockphoto.com

Do pilots use “piloting cases”?



image credit: istockphoto.com

Do parents use “parenting cases”?



image credit: istockphoto.com

Do scientists use “science cases”?



image credit: istockphoto.com

Do managers use “management cases”?



image credit: istockphoto.com

Then why do
testers
use test cases?

One Answer

- Managers like test cases because they make testing “legible”—readable to people who don’t understand testing
- Testers keep talking about test cases because managers keep asking about them.
 - because testers keep talking about them
 - because managers keep asking about them
 - because testers keep talking about them
 - because managers keep asking about them
 - because testers keep talking about them
 - because managers keep asking about them

But smart people
can do MUCH
better than that.

Call this “Checking” not Testing

operating a product to
check specific facts
about it...

means

Observe

Interact with the
product in specific
ways to collect
specific observations.

Evaluate

Apply algorithmic
decision rules to
those observations.

Report

Report any
failed checks.

Three Parts to a Check

1. An *observation* linked to...
2. A *decision rule* such that...
3. they can be both be applied with algorithms—by a **program**.

That means a **check** can be performed



by a machine
that *can't* think
(but that is quick and precise)



by a human who has been
programmed *not* to think
(and who is slow and variable)

image credit: istockphoto.com

Testing Is *More* Than Checking

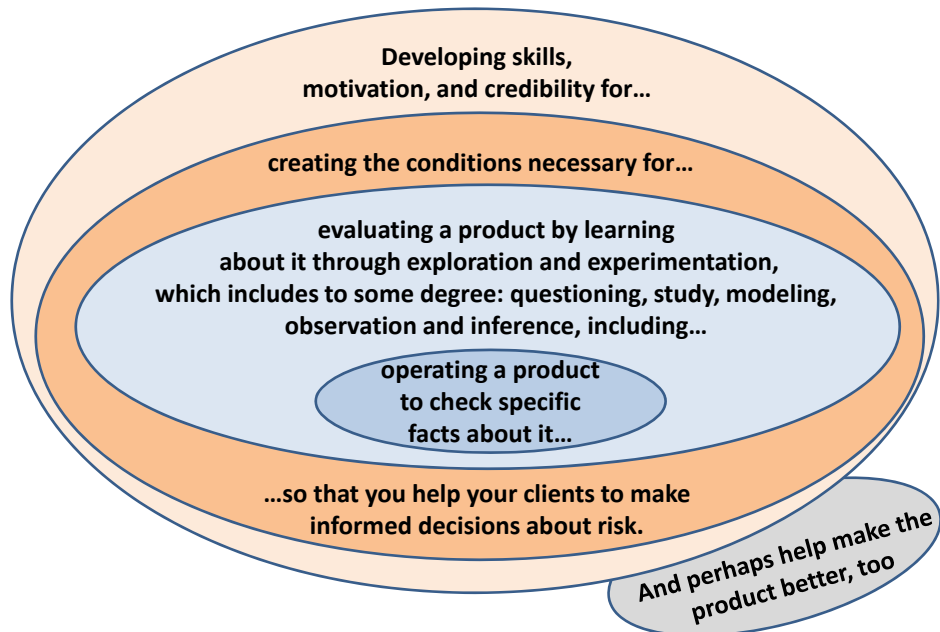
- *Checking* is about confirming and verifying things we *think* are true or *hope* are true
 - But a problem can have MANY problems, even when what we've checked seems correct
 - Checking *can* and *usually should* be done by machines



But I can't THINK!
Who will tell me
how to check?

See <http://www.satisfice.com/blog/archives/856>

Testing is...



A Tale of Four Projects (1)

| Date | Passed | Failed | Total |
|--------|--------|--------|-------|
| 01-Feb | 25 | 225 | 250 |
| 08-Feb | 125 | 125 | 250 |
| 15-Feb | 175 | 75 | 250 |
| 22-Feb | 200 | 50 | 250 |
| 29-Feb | 225 | 25 | 250 |
| 07-Mar | 225 | 25 | 250 |

In Project Blue, a suite of 250 test cases, based on 50 use cases, were written before development started. These cases remained static throughout the project. Each week saw incremental improvement in the code, although things got stuck towards the end.

A Tale of Four Projects (2)

| Date | Passed | Failed | Total |
|--------|--------|--------|-------|
| 01-Feb | 1 | 9 | 10 |
| 08-Feb | 5 | 5 | 10 |
| 15-Feb | 7 | 3 | 10 |
| 22-Feb | 8 | 2 | 10 |
| 29-Feb | 9 | 1 | 10 |
| 07-Mar | 9 | 1 | 10 |

In Project Red, a suite of 10 comprehensive scenarios were constructed and refined as development progressed. In the last week of the project, a change in one of the modules broke several elements in the scenario that worked in the first week.

A Tale of Four Projects (3)

| Date | Passed | Failed | Total |
|--------|--------|--------|-------|
| 01-Feb | 1 | 9 | 10 |
| 08-Feb | 25 | 25 | 50 |
| 15-Feb | 70 | 30 | 100 |
| 22-Feb | 160 | 40 | 200 |
| 29-Feb | 450 | 50 | 500 |
| 07-Mar | 27 | 3 | 30 |

Project Green used an incremental strategy to design and refine test cases. More testers were added to the project each week. As the project went on, the testers also recruited end users to assist with test design and execution.

A Tale of Four Projects (3)

| Date | Passed | Failed | Total |
|--------|--------|--------|-------|
| 01-Feb | 1 | 9 | 10 |
| 08-Feb | 25 | 25 | 50 |
| 15-Feb | 70 | 30 | 100 |
| 22-Feb | 160 | 40 | 200 |
| 29-Feb | 450 | 50 | 500 |
| 07-Mar | 27 | 3 | 30 |

In Week 5 of Project Green, management called a monster triage session that led to the deferral of dozens of Severity 2, 3, and 4 bugs. Nine showstopper bugs remained. Management decreed that only the showstoppers would be tested in the last week.

A Tale of Four Projects (3)

| Date | Passed | Failed | Total |
|--------|--------|--------|-------|
| 01-Feb | 1 | 9 | 10 |
| 08-Feb | 25 | 25 | 50 |
| 15-Feb | 70 | 30 | 100 |
| 22-Feb | 160 | 40 | 200 |
| 29-Feb | 450 | 50 | 500 |
| 07-Mar | 27 | 3 | 30 |

In Week 6 of Project Green, the programmers worked on only the showstopper bugs which were tested using 30 test cases. Testing revealed that six showstoppers are gone, and three persisted. But all the deferred Severity 2, 3, and 4 bugs remained in the product.

A Tale of Four Projects (4)

| Date | Passed | Failed | Total |
|--------|--------|--------|-------|
| 01-Feb | 1 | 9 | 10 |
| 08-Feb | 25 | 25 | 50 |
| 15-Feb | 70 | 30 | 100 |
| 22-Feb | 80 | 20 | 100 |
| 29-Feb | 900 | 100 | 1000 |
| 07-Mar | 900 | 100 | 100 |

In the first few weeks of Project Purple, testers worked interactively with the product to test the business rules, while a team of automation specialists attempted to create a framework that would exercise the product under load and stress conditions.

A Tale of Four Projects (4)

| Date | Passed | Failed | Total |
|--------|--------|--------|-------|
| 01-Feb | 1 | 9 | 10 |
| 08-Feb | 25 | 25 | 50 |
| 15-Feb | 70 | 30 | 100 |
| 22-Feb | 80 | 20 | 100 |
| 29-Feb | 900 | 100 | 1000 |
| 07-Mar | 900 | 100 | 100 |

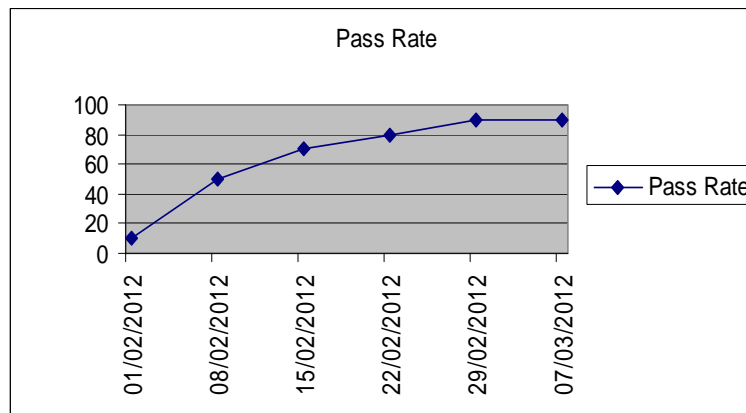
In Week 5 of Project Purple, the automation framework was finally ready. 820 performance scenario tests were run that revealed 80 new bugs, all related to scalability problems. In addition, none of the bugs opened in Week 4 were fixed; two key programmers were sick.

A Tale of Four Projects (4)

| Date | Passed | Failed | Total |
|--------|--------|--------|-------|
| 01-Feb | 1 | 9 | 10 |
| 08-Feb | 25 | 25 | 50 |
| 15-Feb | 70 | 30 | 100 |
| 22-Feb | 80 | 20 | 100 |
| 29-Feb | 900 | 100 | 1000 |
| 07-Mar | 900 | 100 | 1000 |

In Week 6 of Project Purple, the programmers heroically fixed 40 bugs. But there was also a bug in the automation framework, so 40 entirely new bugs were found that week. And they're bad; the programmer report most of them will take at least three weeks to fix.


Four *Completely* Different Projects



...with *exactly* the same pass rate!

Measurement Skills

- What measurement skills can we list that will be useful for testers?
- What tactics can testers use in measurement?
- What issues must testers confront in measurement?
- What IS measurement anyway?



Tell a Three-Part Testing Story

Bugs

A story about the status of the **PRODUCT**...

- ...about what it does, how it failed, and how it *might* fail...
- ...in ways that matter to your various clients.

Oracles

A story about **HOW YOU TESTED** it...

- ...how you operated and observed it
- ...how you recognized problems...
- ...what you have and *have not* tested yet...
- ...what you won't test *at all* (unless the client objects)...

Coverage

A story about how **GOOD** that testing was...

- ...the risks and costs of testing or not testing...
- ...what made testing harder or slower...
- ...how testable (or not) the product is...
- ...what you need and what you recommend.

Issues

image credit: istockphoto.com

Important words!

- **Bug:** any problem that threatens the value of the product
- **Issue:** any problem that threatens the value of the testing, or the project, or the business
- **Coverage:** how much of the product you have tested based on *some model*
- **Oracle:** something that helps you to recognize a problem when it happens during testing.

Hold It!

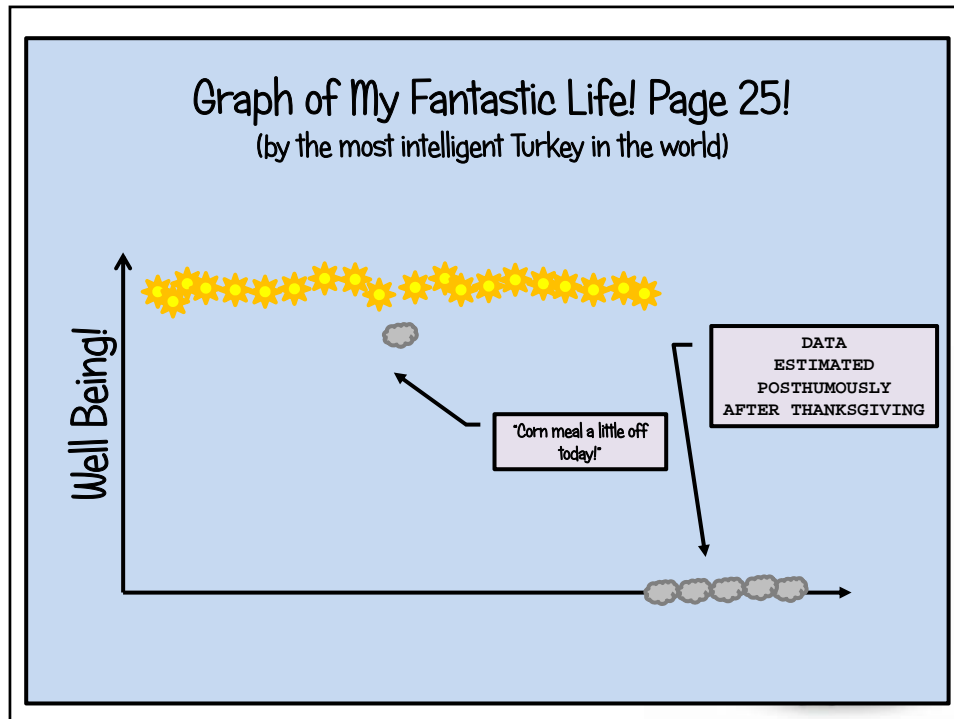
Aren't stories
subjective
by nature?

Sure.

All observations—including quantitative observations—depend on observers, their choices about what to observe, and the quality of their observations.

If testing is telling stories...

- Measurement is a *tool* or *medium* that can help give warrant to those stories.
- Measurement *illustrates* the stories of the quality of the work and the quality of the product.
- Measurement *is not* the story itself.
- As a tool, measurement must be subject to *its* own quality story.
- Tools reflect *what we are*.



Don't Be A Turkey

- No experience of the past can LOGICALLY be projected into the future, because we have no experience OF the future.
- No big deal in a world of stable, simple patterns.
- **BUT SOFTWARE IS NOT STABLE OR SIMPLE.**
- **"PASSING" TESTS CANNOT PROVE SOFTWARE GOOD.**



Based on a story told by Nassim Taleb, who stole it from Bertrand Russell, who stole it from David Hume.

I Don't Hate Numbers

- I love numbers *so much* that I can't stand to see them abused as they are by people in our profession.
- This workshop is designed to take you deeper into measurement, spotting critical thinking errors that might cause you to miss observations and mislead your client—or yourself.
- The intention is not to suggest that measurement is useless, but to expand our notions of what measurement might be.

Imperfections in measurement are always a problem, but they're a devastating problem only when we don't recognize them.

—Daniel Gilbert, *Stumbling on Happiness*

What Is Measurement?

Measurement is the art and science of making reliable observations.

—Jerry Weinberg, *Quality Software Management Vol. 2*

- Since the time of Aristotle (at least), we've known about two kinds of measurement that inform decisions
 - “Two pounds of meat”
 - “Too much”, “too little”, “just right”.

We waste time and effort when we try to obtain six-decimal-place answers to whole-number questions.

- <http://www.developsense.com/articles/2009-05-IssuesAboutMetricsAboutBugs.pdf>
- <http://www.developsense.com/articles/2009-07-ThreeKindsOfMeasurement.pdf>
- <http://www.developsense.com/articles/2007-11-WhatCounts.pdf>

Questioning Measurement

What might we observe?

- What are we choosing NOT to observe?
- What are we not observing by accident?
- What comparisons can we make?
 - are those comparisons valid? relevant?
 - are those comparisons reliable?
- What are we trying to assess?
 - How will we respond to the information?
 - Will we use it to control? Or to learn?

The quality of our measurement depends upon our skill at observation, what we're comparing, and the validity of the models that we're using for assessment.

Why Do We Measure?

- self-assessment and improvement
- evaluating project status
- evaluating staff performance
- informing others about the characteristics of the product
- informing external authorities about the characteristics of the product

—Kaner and Bond

See <http://www.kaner.com/pdfs/metrics2004.pdf>
See Robert Austin, *Measuring and Managing Performance in Organizations*

But there's an over-riding reason...

Fear of irrationality

Quantifying something complex provides a kind of observational integrity. Expressing things in numbers comes with a side effect: *information loss*. As Cem Kaner describes the social sciences, think of testing and measurement as providing *partial answers that might be useful*.

Ignore the risks of measurement, and distortion and dysfunction are likely. Manage to your metric, and dysfunction is *guaranteed*.

What Is Measurement?

“Measurement is the empirical, objective assignment of numbers, according to a rule derived from a model or theory, to attributes of objects or events with the intent of describing them.”

- Source: “Software Engineering Metrics: What Do They Measure and How Do We Know?”
 - Cem Kaner and Walter P. Bond
 - <http://www.kaner.com/pdfs/metrics2004.pdf>

What happens when we walk through that definition?

How Do We Measure?



- Third-order measurement
 - highly instrumented, used to discover natural laws
 - “What *will* happen? What *always* happens?”



- Second-order measurement
 - often instrumented, used to refine first-order observation
 - used to tune existing systems
 - “What’s *really* going on here? What’s happening right now?”

How Else Do We Measure?



- First-order measurement
 - minimal fuss, direct observation, minimal instrumentation
 - used to inform a control action OR to prompt search for more refined information
 - “What’s going on? What should we do? Where should we look?”

Weinberg suggests that, in software development, we’re obsessed with trying to make third- and second-order measurements when first-order measurements might be all we need—and tend to be much cheaper.

A Synthesis

“Measurement is the art and science of making reliable observations, applying values based on modeling and comparison of objects, attributes, or events, for the purposes of understanding distinctions, making assessments, and informing evaluations.”

- My intention here is to highlight
 - why we measure (assessment and evaluation)
 - how we measure (observation, and comparison based on models)
- Measurement might be qualitative or quantitative, but assessment and evaluation are always qualitative:

What do we want?

Why Prefer First-Order Measures?

- When you're driving, are you mostly concerned about...
 - your velocity, acceleration, vehicle mass, drag co-efficient, frictional force? (third-order)
 - your engine temperature, RPMs, and current rate of gas consumption? (second-order)
 - looking out the window to avoid hitting things (first-order)?



I've observed *many* projects that have crashed because managers were overly focused on the dashboard instead of the traffic and obstacles around them, and the road ahead.

What kind of driver do you trust?

Possibly Useful First-Order Measures

- The Binary Release Metric
 - “Any showstoppers?” (A showstopper is a problem that makes more sense to fix than to release.)
 - IF answer == YES, then don’t ship
- The Issues List
 - list bugs and issues by importance to some stakeholder; optionally rank them first
 - note that testability issues may be most important; problems that prevent or slow down testing mean that other problems have more places to hide
- The Furrowed Brow Test (a.k.a. Squirm Test)
 - announce that we’re planning to ship on schedule
 - observe posture, furrowed brows, and squirming

Measurement vs. Metrics

- A *measurement* is an observation that relates on thing to another, based on some model or theory, for the purpose of making some distinction.
- The *metric* is the operation and the mathematical function that establishes or (or “maps”) the relationship.



A

<



B

Theory:

**If $\text{len}(A) < \text{len}(B)$
then A is not really a
foot-long dog.**

Measurement Scales

equality (☞ identity; constructs; this IS that)

nominal (☞ labels; the number isn't quantity at all)

ordinal (☞ order; number suggests a rank)

interval (☞ differences, and equalities in them)

ratio (☞ differences and proportions in differences)

derivative (☞ combinations of the above)

**Mistake one kind of scale for another, and
you can guarantee invalid, unsupportable conclusions.**

See S.S. Stevens, "On the Theory of Scales of Measurement", *Science*, June 17, 1946

Nominal Scale



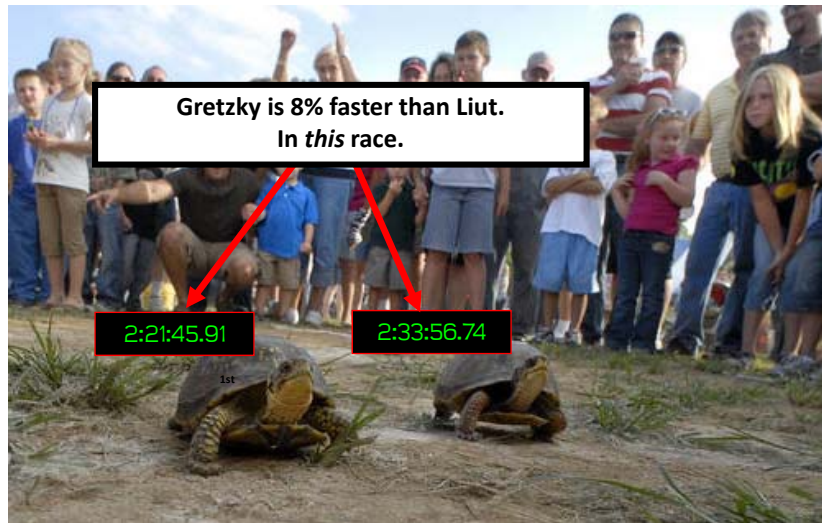
Ordinal Scale



Interval Scale



Ratio Scale



What Statistics Can You Do for Each Scale?

- **Nominal**
 - assign labels
 - count an instance of a construct, the total instances of each, the grand total, the greatest number of instances (a.k.a. the mode)
- **Ordinal**
 - order by rank
 - identify the median (the middle element), divide into percentiles
- **Interval**
 - count units of distance between items
 - doesn't require a true zero point: think calendars, thermometers
 - identify the mean, calculate the standard deviation, identify why they've earned a given rank, etc. –but no zero-based stats
- **Ratio**
 - find multiples of distance between items; magnitudes
 - *does* require a true zero point
 - includes *fundamental* and *derived* measures

Three Important Kinds of Reliability

- Quixotic reliability
 - the measurement yields consistent results in different circumstances
 - could be the result of a broken instrument, “party line” answers
 - Diachronic reliability
 - the measurement yields the same result when taken multiple times
 - only reliable for things that don't change in a changing world; “may deny history”
 - Synchronic reliability
 - similarity of observations within the same time period
 - reveals potentially significant questions when it fails
- Kirk and Miller

Two Important Kinds of Validity

- Construct validity: to what degree is what you're measuring an instance of what you think you're measuring?
 - The “counting to one” problem
- External validity: to what degree are can your measurements support generalizations to cases outside your study?
 - The “so you measured; so what?” problem

Measurement without consideration of validity is playing with numbers.

Kaner & Bond on Measurement Validity

1. What is the purpose of your measurement?
2. What is the scope of the measurement?
3. What is the attribute you are trying to measure?
4. What are the scale and variability of this attribute?
5. What is the instrument you're using
6. What are the scale and variability of the instrument ?
7. What function (metric) do you use to assign a value to the attribute?
8. What's the natural scale of the metric?
9. What is the relationship of the attribute to the metric's output?
10. What are the natural, foreseeable side effects of using this measure?

The essence of good measurement is a model that incorporates answers to questions like these.

If you don't have solid answers, you aren't doing measurement; you are just playing with numbers.

Darrell Huff's Five Questions

- Who says so?
- How do they know?
- What's missing?
- Did somebody change the subject?
- Does it make sense?

From *How to Lie with Statistics*, originally published in 1954.

The Power of Limited Information

- Snap judgments and heuristics are central to our decision-making process
- Because they're fast and frugal, they may sometimes be more valuable than complex and rigorous analysis

People often make snap (first-order) observations or decisions and then use (second-order) numbers to *test* them.

Control vs. Inquiry Measurement

- A **control measurement** is a measurement that **drives decisions**.
 - Any measurement you use to control a self-aware system will be used by that system to control YOU.
- An **inquiry measurement** is any measurement that **helps you ask the right questions at the right time**.
 - Inquiry measurements are also vulnerable to gaming, but the stakes are far lower, so there's less incentive for manipulation.




Text here is taken from the work of my colleague, James Bach.
<http://www.satisfice.com>

Control vs. Inquiry

- Remove *control metrics* that are linked to pay, bonuses, performance evaluation, etc.
 - control metrics trigger some action, usually automatically
 - a metric that is used to control something will eventually be used to control you
- Foster *inquiry metrics*
 - inquiry metrics prompt us to ask questions
- Relax measurement when the metric stops changing
 - if you're not obtaining new information, try measuring something else of interest for a while

Counting Bugs? Or Identifying Issues?

- In software testing we talk about finding bugs, and reporting them.
 - There's something worse than a bug: an *issue* —something that slows down or prevents your ability to find a problem.
 - Mechanisms intended to provide management information often *ignore* issues.
- 
- Bug reporting systems focus on bugs, and suppress reporting of issues
 - Mechanisms intended to provide management control often *create* issues.
 - For a description of issues and how we might report them, see <http://www.developsense.com/blog/2011/01/youve-got-issues/>.

Lines of Code

Here are two lines of code from the same program.
Are they equivalent?

```
obj.visibility=v;
for(i=0;!x&&d.layers&&i<d.layers.length;i++)
\ x=MM_findObj(n,d.layers[i].document);
```

In the first example, it appears as though one bit is being set.

In the second, multiple values are (conditionally) being initialized, compared, set, incremented, referenced, or dereferenced.

This is like counting tricycles and space shuttles as equivalent items.

What Are The Factors of a “Test Case”?

Power: will this test reveal a problem?

Validity: is problem revealed a genuine problem?

Value: is the information is useful to your product, project, or client?

Credibility: will clients believe it represents things people will actually do?

Representative: is it a good example of plausible similar tests?

Motivating: will the test make us want to fix the problem it reveals?

Performable: Can the test be performed as designed?

Maintainable: Can the test be revised easily when the product changes?

Reusable: It is easy and inexpensive to reuse the test for other products?

Pop: Will the test challenge our assumptions and reveal new information?

Coverage: Will the test exercise the product in a unique way?

Easy to evaluate: Is there a clear answer to the question the test poses?

Many of these ideas come from
Kaner & Bach’s Black Box Software Testing Course

<http://www.wtst.org/WTST7/BBSTwtst2008kanermeeting.pdf>

What Are The Factors of a “Test Case”?

Supports debugging: Will it provide useful results for the programmer?

Repeatable: does the test reveal a problem consistently?

Mutability: can the test be adapted to other test ideas?

Complexity: are there interesting interactions between components?

Simplicity: does the test successfully isolate a problem of interest?

Accountability: can you explain, justify, and prove that you run the test?

Equipment cost: do you need specialized gear to run the test?

Development cost: what resources are required to design the test?

Setup cost: what time and resources are required to prepare for the test?

Execution time: how long does it take the test to run?

Reporting time: what effort is required to communicate the results?

Opportunity cost: what valuable work could you do instead of this test?

Many of these ideas come from
Kaner & Bach’s Black Box Software Testing Course

<http://www.wtst.org/WTST7/BBSTwtst2008kanermeeting.pdf>

Surrogate Measures

- We don't know how to measure
 - the quality of the product
 - the quality of testing
 - the quality of a tester
- So we count something easily countable
 - Requirements
 - Bug counts
 - Test cases
 - Test cases per requirement
 - Test cases per day
 - Defect escapes
- These are called *surrogate measures*
- Do they measure quality?

Don't Just Count Them!

- You've just seen at least 24 factors by which we might describe or evaluate a given test
- Bugs have similar numbers of factors, if only we pause to think about them
- Many factors aren't usefully quantifiable
 - yet they might be supremely important
 - people base decisions on politics and emotions
 - people have emotional reactions to software
- Models may leave out many dimensions
 - some of which might also be very important
- *Testers* are even more complex
 - tester effectiveness needs multi-dimensional measures

Counting *Ideas*

- Don't count test cases
 - test cases are *ideas*; ideas aren't things
 - counting *ideas* is reification error
- Don't judge product health by bug counts
 - try emphasizing bug stories
- Don't measure testers by counting bug reports
 - testers may be doing other things of great value besides writing bug reports

"If you evaluate testers by counting bug reports, I *guarantee* that your testers are misleading you."

"Test cases are like briefcases; I'm a 'famous testing expert', and I can't tell whether 1000 test cases are good or bad until I see them.

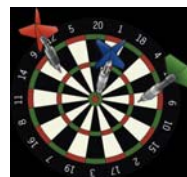
James Bach

Precision vs. Accuracy

- What time is it, *exactly*, *right now*?
- My watch say 5:35:21, which is precise, but it's *wrong*.
- Many people will offer precise numbers—but will they be *accurate*?



Precise!



Accurate!

Heuristic: In software development processes, the more precise the number provided, the more likely it is to be based on an unsupportable model.

Let's Get Serious

- If our numbers are
 - multivariate
 - based on incomplete models
 - rough, first-order approximations

Let's not overwhelm ourselves
wasting time and money
seeking bogus precision.

Quantifying vs. Qualifying

- Comparisons and assessments aren't necessarily numerical (ask Goldilocks).
- Numbers aren't as descriptive as words and stories.
- Words can be vague or ambiguous, but numbers without clarifying words are just as bad or worse.
- Could you tell convincing, motivating stories?
- Could you use ranking or laddering?
- Could you use reports and opinions from multiple people?

What if you asked for
"things you liked
and things you didn't like"?

Assessment Without Numbers

- Observation can go directly to assessment without quantified measurement
 - i.e. first order measurement
- Ask what other modes, beside numerical ones, you could use for evaluation
 - start by asking *what problem you want to solve* or *what situation you'd like to assess*
- If you're worried that observations and assessments are subjective, ask *several* people who matter
- The object of the game is usually NOT mathematical precision; it's reliable observation

Qualifying Coverage

- We can compose, edit, narrate, and justify *stories* about our coverage
- We can use product elements, quality criteria, test coverage outlines, and risk lists to frame those stories
- We can even quantify coverage *with respect to a particular model* if we provide and illustrate a clear story about the model

Test Coverage

- The amount of the product that has been tested with respect to coverage criterion (that is, some idea about completeness).
- Model the product
- Cover the map!
 - “Maps” can be lists, tables, matrices, diagrams, graphs, mind maps,...

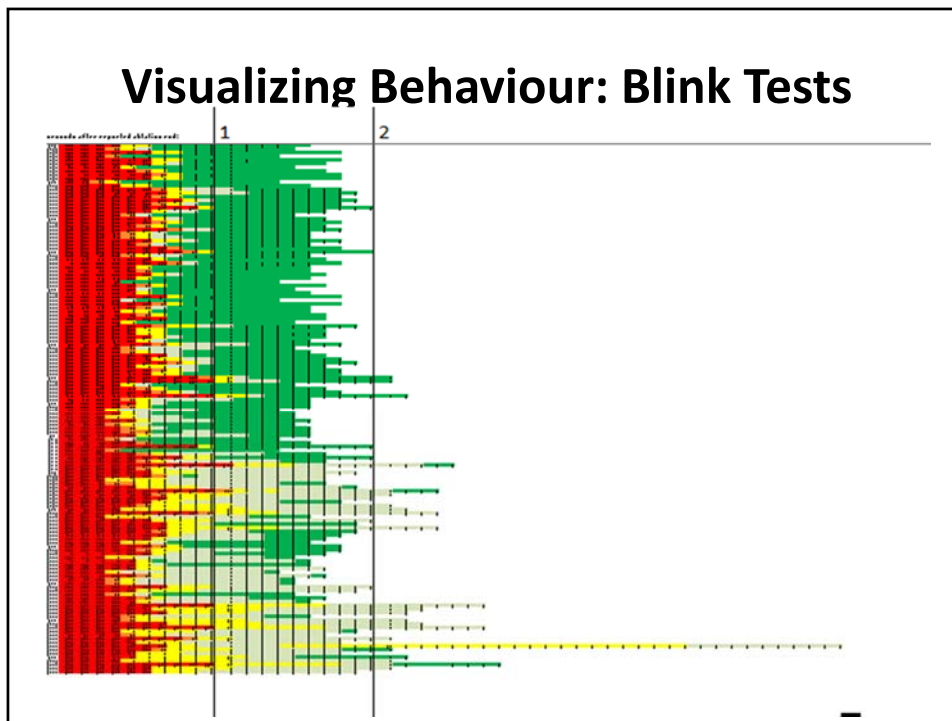
Extent of Coverage

1. Smoke and sanity
 - Can this thing even be tested at all?
2. Common, core, and critical
 - Can this thing do the things it *must* do?
 - Does it handle happy paths and regular input?
 - *Can* it work?
3. Complex, harsh, extreme and exceptional
 - Will this thing handle challenging tests, complex data flows, and malformed input, etc.?
 - *Will* it work?

Quality Criteria Coverage Matrix

| Test Idea | Correctness | Reliability | Usability | Security | Maintainability | Performance | Interactivity | Comprehensibility | Supportability | Flexibility | Portability | Installability |
|--|-------------|-------------|-----------|----------|-----------------|-------------|---------------|-------------------|----------------|-------------|-------------|----------------|
| Notepad can exchange data with any other Windows application via cut and paste. | X | | | X | | | | | | | | |
| Notepad has some kind of user documentation. | | X | | | | | | | | | | |
| The title bar font selected in the Appearance section of the Display control panel is properly reflected in Notepad. | | | | | | | | | | | | |
| Search function works with a range of files, big and small, and a range of search strings. | | | | | | | | | | | | |
| Notepad can print text to the default printer. | | | | | | | | | | | | |
| By default, Notepad will be located on the Accessories sub-menu of the Vista Start menu. | | | | | | | | | | | | |
| Notepad never crashes. | | | | | | | | | | | | |
| Notepad can be run alongside many other applications on the same desktop. | | | | | | | | | | | | |
| Many instances of Notepad can be executed simultaneously, without performance degradation. | | | | | | | | | | | | |
| Double-clicking on any text file icon will automatically launch Notepad. | | | | | | | | | | | | |
| Notepad doesn't take too much space on the disc, and is easy to remove. | | | | | | | | | | | | |
| Time/Date stamp feature properly inserts system time and date in a wide variety of situations. | | | | | | | | | | | | |
| No memory leaks occur that cause Notepad to fail. | | | | | | | | | | | | |
| Notepad is included in the default Windows installation. | | | | | | | | | | | | |
| Notepad takes as little or less time to launch than any other application on the same platform. | | | | | | | | | | | | |
| TOTAL | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Visualizing Behaviour: Blink Tests



Visualizing Estimation

- project size (obvious size vs. hidden)
- project complexity
- development staff (people/skills)
- development resources (time, in-kind collaboration)
- desired quality level
- availability of requirement and design information
- availability
- routine information (meetings, email, etc.)
- resources
- extraordinary information
- accounting and auditing
- change control processes
 - change
 - scope
 - configuration
 - tools
- test tooling (software, needed testing)
- test administration (status reports, documentation, coordination cost)
- tester training and learning curve
- test data preparation
- test environment (reliability, installation, stability)
- test tooling
- the time scale of the estimate
 - required accuracy
 - measurement
- regression or other dependencies
- relationship between system and client
- time required to prepare the estimate
- revision triggers
- number and significance of bugs
- completion and reporting when done testing
- other by development work
- specific new features and design changes
- driven by business needs
- implementation of multiple projects for test cycles
- specific case risks
 - for those that have failed awaiting their own recovery
- local dependencies
 - program's "back door" to test its health?
 - vendors may be waiting on programmers
- third-party dependencies
 - vendors can't build until we're ready?
 - programmers or testers may be waiting on customer
- implementation (testing sequencing and delays)
 - when doing delays, test in what order
 - how quickly can the system be tested?
- specific dependencies
 - hardware and black boxes
- competing projects
 - when our project's schedule slips, what will happen?

In this list, testing has

- plenty of control over green factors;
- some control over yellow factors, and
- very little or none over the red ones.

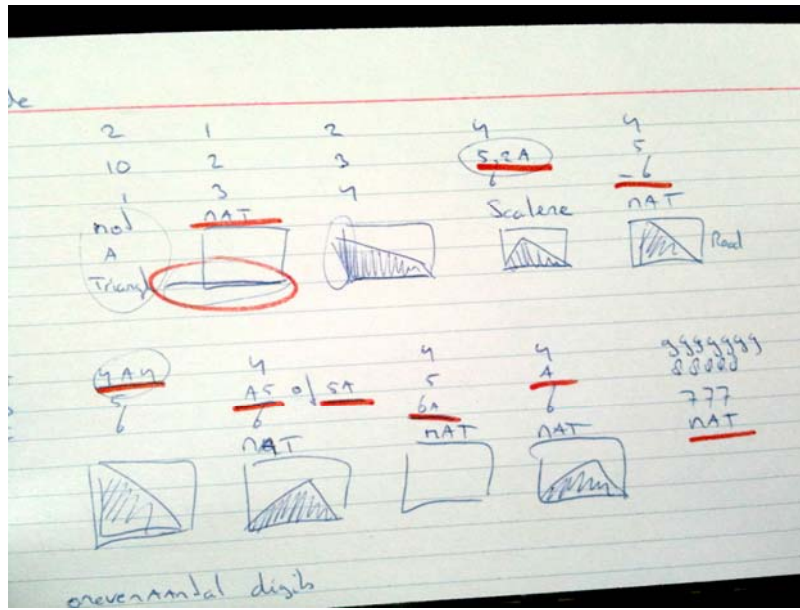
So why ask testers for an estimate?

Estimation

- Question assumptions
 - Testing is a responsive activity
 - Think of prospecting: we don't know when we're going to find the gold
 - Most people, when estimating the test cycle, are really estimating the *fix* cycle
 - It could continue forever
 - It can be stopped at any time
- Ask "When do you want to ship?"
 - that's how much time you have, *and need*, to test

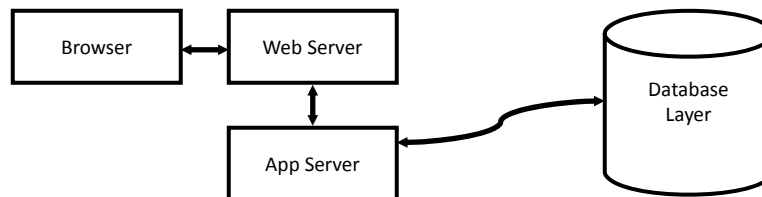
<http://www.developsense.com/blog/2010/10/...project-estimation-and-black-swans-part-5-test-estimation>

Visualizing Test Results



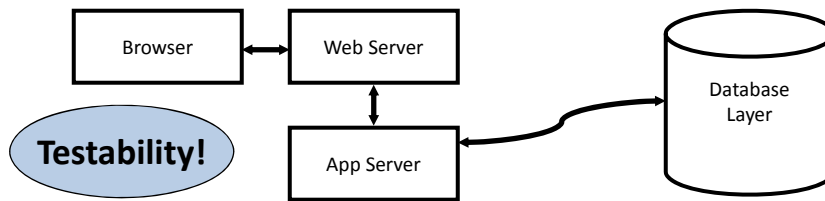
Visualizing Risk: Analysis

- [pointing at a box] *What if the function in this box fails?*
- *Can this function ever be invoked at the wrong time?*
- [pointing at any part of the diagram] *What error checking do you do here?*
- [pointing at an arrow] *What exactly does this arrow mean? What would happen if it was broken?*

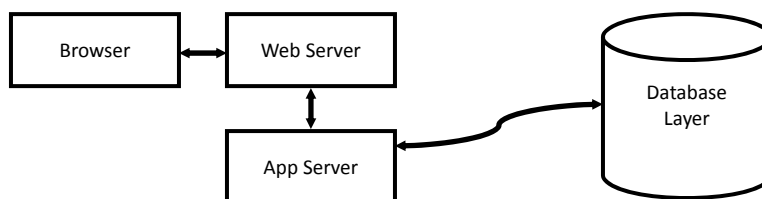








Guideword Heuristics for Diagram Analysis

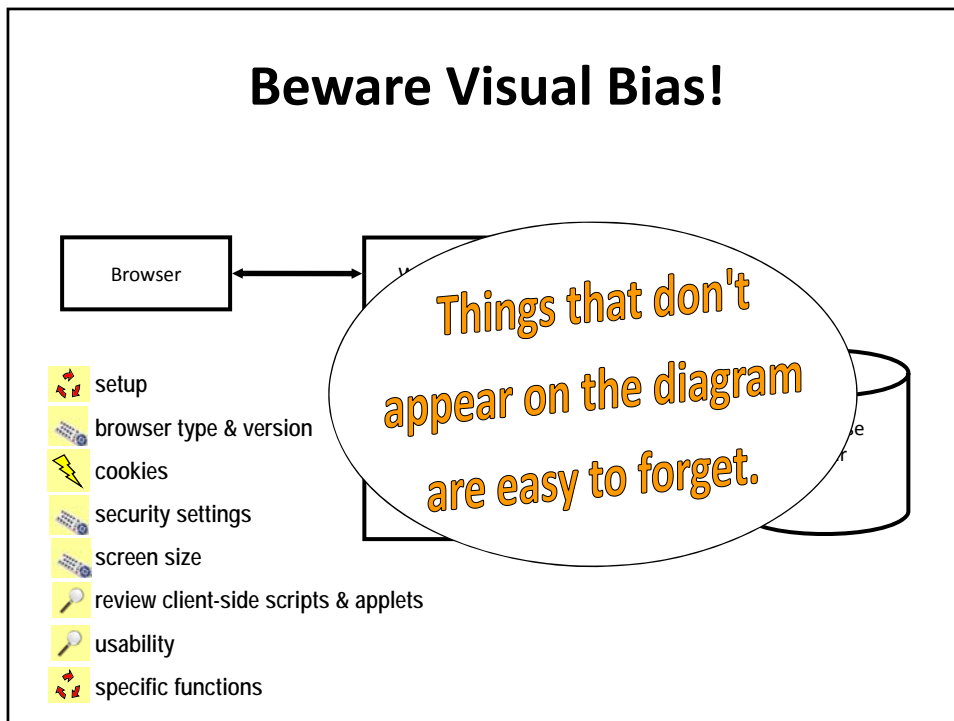
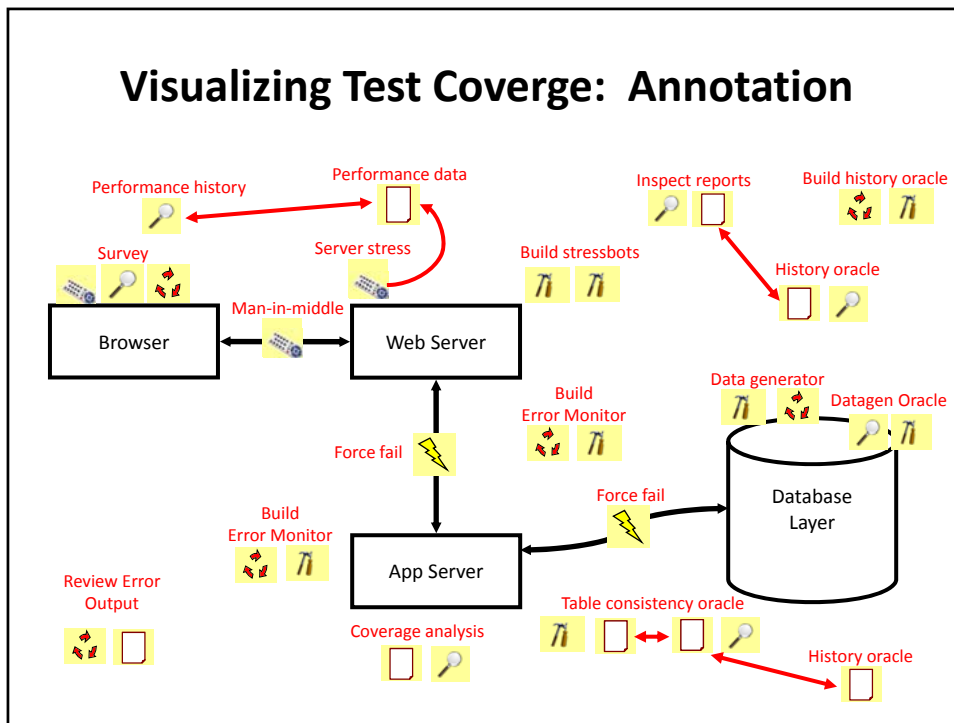
- Boxes
 - Missing/Drop-out
 - Extra/Interfering
 - Incorrect
 - Timing/Sequencing
 - Contents/Algorithms
 - Conditional behavior
 - Limitations
 - Error Handling
- Lines
 - Missing/Drop-out
 - Extra/Forking
 - Incorrect
 - Timing/Sequencing
 - Status Communication
 - Data Structures
- Paths
 - Simplest
 - Popular
 - Critical
 - Complex
 - Pathological
 - Challenging
 - Error Handling
 - Periodic



Visualizing Test Coverage: Annotation



- | | | | |
|---|-----------------------|---|----------------|
|  | Observation or Oracle |  | Tool Focus |
|  | Control/Vary/Modify |  | Data or Object |
|  | Force Fail |  | Activity |



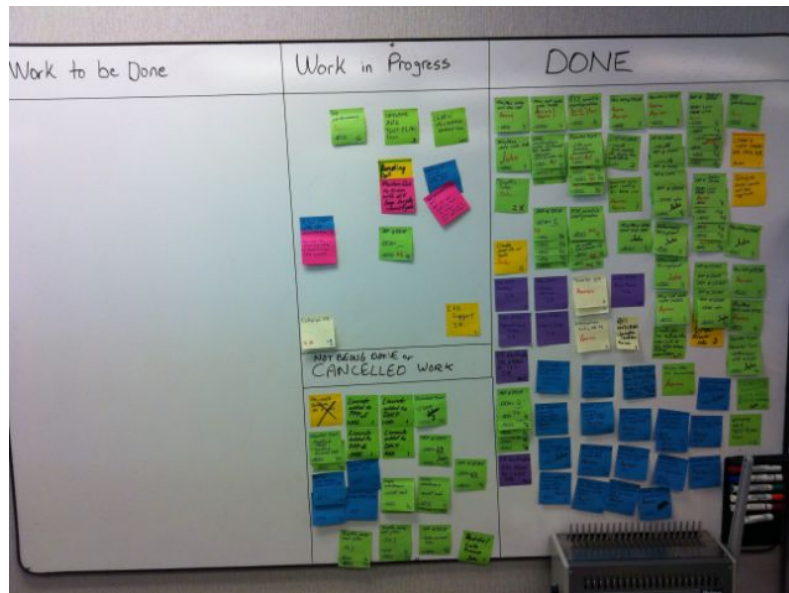
Visualizing Test Progress



Visualizing Test Progress



Visualizing Test Progress



Why Is Testing Taking So Long?

- Try tracking time spent on
 - test design and execution
 - bug investigation and reporting
 - test setup
 - none of the above (non-testing work such as meetings, email, administrivia, time-tracking, waiting, etc.)
- Try tracking this to 5% - 10% granularity
 - finer granularity means that “time tracking” requires significant non-testing effort
 - do you include “time spent on time tracking” in your time tracking?
 - do you include “time to estimate” in their estimates?

Test Session Effectiveness

- A “perfectly effective” testing session is one entirely dedicated to test design, test execution, and learning
 - a “perfect” session is the exception, not the rule
- Test design and execution tend to contribute to test coverage
 - varied tests tend to provide more coverage than repeated tests
- Setup, bug investigation, and reporting take time away from test design and execution
- Suppose that testing a feature takes two minutes
 - this is a highly arbitrary and artificial assumption—that is, it’s *wrong*, but we use it to model an issue and make a point
- Suppose also that it takes eight extra minutes to investigate and report a bug
 - another stupid, sweeping generalization in service of the point
- In a 90-minute session, we can run 45 feature tests—*as long as we don’t find any bugs*

How Do We Spend Time? (assuming all tests below are *good* tests)

| Module | Bug reporting/investigation (time spent on tests that find bugs) | Test design and execution (time spent on tests that find no bugs) | Number of tests |
|----------|---|--|-----------------|
| A (good) | 0 minutes (no bugs found) | 90 minutes (45 tests) | 45 |
| B (okay) | 10 minutes (1 bug, 1 test) | 80 minutes (40 tests) | 41 |
| C (bad) | 80 minutes (8 bugs, 8 tests) | 10 minutes (5 tests) | 13 |

Investigating and reporting bugs means....

SLOWER TESTING or...
REDUCED COVERAGE ...or both.

- In the first instance, our *coverage* is great—but if we’re being assessed on the number of bugs we’re finding, we look bad.
- In the second instance, coverage looks good, and we found a bug, too.
- In the third instance, we look good because we’re finding and reporting lots of *bugs*—but our *coverage* is suffering severely. A system that rewards us or increases confidence based on the number of bugs we find might mislead us into believing that our product is well tested.

What Happens The Next Day?

(assume 6 minutes per bug fix verification)

| Fix verifications | Bug reporting and investigation today | Test design and execution today | New tests today | Total over two days |
|-------------------|---------------------------------------|---------------------------------|-----------------|---------------------|
| 0 min | 0 | 45 | 45 | 90 |
| 6 min | 10 min (1 new bug) | 74 min (37 tests) | 38 | 79 |
| 48 min | 40 min (4 new bugs) | 2 min (1 test) | 5 | 18 |

Finding bugs today means....

VERIFYING FIXES LATER

...which means....

EVEN SLOWER TESTING or...
EVEN LESS COVERAGE ...or both.

- ...and note the optimistic assumption that all of our fixed verifications worked, and that we found no new bugs while running them. Has this ever happened for you?

Testing vs. Investigation

- Note that I just gave you a compelling-looking table, using simple measures, but notice that we still don't know anything about...
 - the quality and relevance of the tests
 - the quality and relevance of the bug reports
 - the skill of the testers in finding and reporting bugs
 - the complexity of the respective modules
 - luck

...but if we *ask better questions*, instead of letting data make our decisions, we're more likely to make progress.

Principles for Data Presentation from Edward Tufte

- The point of display is to aid thinking
 - What is the thinking task?
- Data is about relationships; show comparisons
- Where causality is known, show causality
 - where causality isn't know, seek patterns
- Show more than one variable and changes in the variables
 - which may expose previously unnoticed patterns
- Integrate word, number and image
 - “don't make a report replicate the chaotic mode of production”

from notes taken at Tufte's one-day class, Dearborn, MI, 2004

Principles for Data Presentation from Edward Tufte

- Documentation quality shapes credibility
 - make the source of the data part of the display
- Show as many (or as long) intervals adjacent in space
 - not stacked in time
- Use as large a display as possible
 - large paper, big monitors, multiple monitors
- Use small multiples
 - use the same scale consistently

from notes taken at Tufte's one-day class, Dearborn, MI, 2004

Principles for Data Presentation from Edward Tufte

- At least six dimensions available: height, width, length, time, revealed/concealed, and colour
- More data horizontally provides the usual basis for comparison
- Design of information usually recapitulates
 - the corporate hierarchy
 - computer systems
- Visual metaphors should be based on the content

from notes taken at Tufte's one-day class, Dearborn, MI, 2004

Break Down the Testing Story

- Who are the characters? (People? Bugs?)
- What matters to them?
- What happens over time?
- Why do they do what they do?
- When did the story happen?
- Where did they go?
- What did they do?

Why should we care?

Tell The Testing Story

For a test cycle, be prepared to describe...

- the testing mission
- specific risks to address
- the diversity of your coverage, oracles, and techniques
- what you might be missing, and why it's okay

Defocusing!

Try Other Modes of Assessment

- Try private chats, standup meetings, or scrums
 - short meetings that identify needs for further meetings between smaller groups, to lessen wasted time
- Try laddering exercises
 - ranking, rather than measuring
 - if they're influenced by feelings, that may be OK
 - emotions are signals; look into them
- Try *talking to people*
 - try asking for descriptions instead of numbers
 - try retrospectives

Reporting

- “Never give a number to a bureaucrat”
 - Plum’s Second Law
- Emphasize stories and narratives
- Don’t produce, offer or accept a number without a story
 - lead with the story
 - show the multivariate nature of data
 - annotate charts or tables
 - note several possible interpretations
- Prefer direct observations and simple comparisons over derivative or surrogate measures

Three Heuristics

*The numbers are not the story.
Seek information, not just data.
Always ask “Compared to what?”*

Summary

- Testing is about noticing
- What haven't you noticed today?
- Measurement is **not** analysis
- Measurement is *a tool for analysis*

“We shape our tools;
thereafter, our tools shape us.”

---Marshall McLuhan

Finale: A Useful Measurement

- Weinberg's Management Measurement
 - “How preoccupied the managers are with overly precise measurements for which they have no working models”

If you watch where people are looking, it can tell you something about where they're *not* looking.

That may be where the biggest risks live.

For a valid *predictive* software development metric, you need **a stable team and a stable project.**



Readings

- *Quality Software Management, Vol. 2., "First Order Measurement"*
- *Exploring Requirements: Quality Before Design*
 - Gerald M. Weinberg
- *"Software Engineering Metrics: What Do They Measure and How Do We Know?"*
 - Cem Kaner and Walter P. Bond
 - online at <http://www.kaner.com/pdfs/metrics2004.pdf>
- *How to Lie with Statistics*
 - Darrell Huff
- *The Black Swan*
- *Fooled by Randomness*
 - Nassim Nicholas Taleb

Readings

- *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*
 - Shadish, Cook, Campbell
- *Reliability and Validity in Qualitative Research*
 - Kirk & Miller
- *Tools of Critical Thinking: Metathoughts for Psychology*
 - David Levy
- *"From Neurons to Neighborhoods: The Science of Early Childhood Development"*
 - Shonkoff and Phillips,
<http://www.nap.edu/openbook.php?isbn=0309069882>
- *How to Measure Anything: Finding the Value of "Intangibles" in Business*
 - Douglas Hubbard

Readings

- *Visual Explanations*
 - Edward Tufte
- *Stumbling on Happiness*
 - Daniel Gilbert
- *Why Does Software Cost So Much?*
 - Tom DeMarco
- *What's Your Story?*
 - Craig Wortmann
- *Measuring and Managing Performance in Organizations*
 - Robert D. Austin
- *Freakonomics*
 - Stephen Leavitt
- *Perfect Software and Other Illusions About Testing*
 - Gerald M. Weinberg

Quick Summary

- Measurement: of the talks, books, and articles on metrics you find, what proportion fail to mention validity?
- Measurement: in discussions on measurement, what proportion fail to distinguish between "measurement" and "metric"?
- Try measuring this: in presentations on measurement, what proportion say "It's not easy" without providing any how-tos?
- *Measurement is the art and science of making reliable observations.*
- A metric is a measurement function that relates an observation with a number. --Cem Kaner
 - A number or a value.
- Construct validity involves questioning the function relating an observation and a number. Thus construct validity also includes questioning our observations and our classifications.