Who Does Quality Assurance?

Michael Bolton
DevelopSense
http://www.developsense.com
April, 2011

Updates

- This presentation is ALWAYS under construction
- Updated slides at http://www.developsense.com/past.html
- All material comes with lifetime free technical support

I'm a consultant. I specialize in solving testing problems that other people can't solve—and helping them learn how to do that for themselves. You can read more about that at http://www.developsense.com

This presentation, like all of my work, comes with lifetime free technical support. I'm delighted to received follow-up questions, comments, disagreements, alternative interpretations, amplifications, and stories. Drop me a line at michael@developsense.com; call me at +1 (416) 656-5160; reach me on Twitter @michaelbolton. I'm often found in other places too, but those are where messages will tend to get to me most quickly.

Michael Bolton

http://www.developsense.com

michael@developsense.com

I'm a software tester, consultant, and trainer with 20 years of experience around the world, testing, developing, managing, and writing about software. I am the co-author (with senior author James Bach) of Rapid Software Testing, a course that presents a methodology and mindset for testing software expertly in uncertain conditions and under extreme time pressure.

Currently, I lead DevelopSense, a Toronto-based consultancy. Prior to that, I was with Quarterdeck Corporation for eight years, during which I managed the company's flagship products and directed project and testing teams both in-house and around the world. And even deeper in the past, I was a stage manager for a professional school touring theatre company. (You have no idea how dealing with actors, children, schoolteachers, principals, and janitors prepares one for dealing with programmers, business people, marketers, and executives!)

Let’s Start With a Simple Question:

What is "quality"?

Is Quality Conformance to Standards?

Is Quality “Conformance to Expectations”?

Windows Vista™

- I *expected* it to suck.
- It sucked.

Quality Answers

- Quality is “value to some person(s)”

  – Jerry Weinberg

- “…who matter.”

  – James Bach and Michael Bolton

- Decisions about quality are always political and emotional
  - made by people with the power to make them
  - require a determination of *who matters*
  - made with the desire to *appear* rational
  - yet ultimately based on how those people *feel*

If you're *not* a manager, do you…

approve design decisions?

sign the customer contracts?

settle disagreements?

hire staff?

decide which problems to fix?

allocate staff?

set the schedule?

set the product scope?

fix project problems?

decide on raises?

allocate training budgets?

declare priorities?

choose the production model?

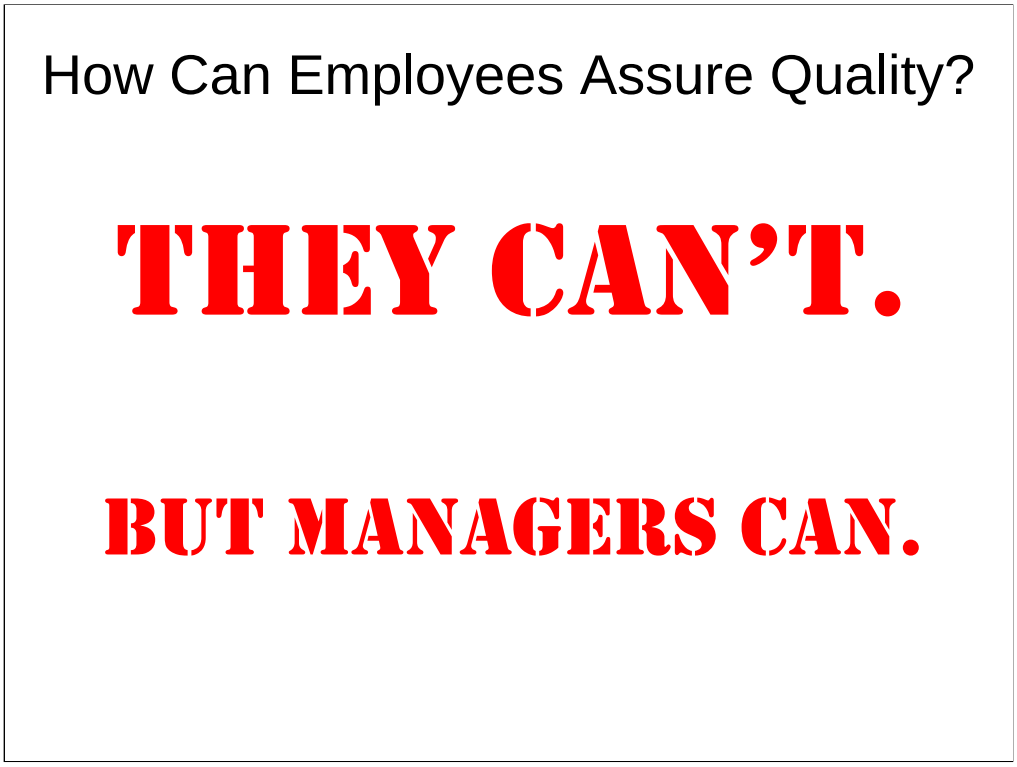
fire some staff?

control the budget?

set the company's strategic direction?

Can employees answer Yes to *all* of those questions? Any? No?

Then how, exactly, do employees ASSURE quality?



Actually, that's not *strictly* true. Employees *can* assure the quality of *their own* work, and by and large, they're intrinsically motivated to do it.

What employees *can't* do, though, is to assure the quality of the product and project *overall*.

Even though ideas like Lean manufacturing and Agile development propose group decision-making, I've never seen more than a couple of names on a cheque or on hiring approvals. Consensus is really important. It's a foolish manager that ignores his people, but final decisions still rest with some person.

How Can Managers Assure Quality?

At least five things that you *must* do:

1. Find out who matters.
2. Note where the buck stops (with you).
3. Understand the work being done.
4. Provide the needed resources.
5. Clear the obstacles.

To do all this well,
you must TEST and MEASURE.

From my experience as a program manager for a commercial software company, I argue that there are five things that managers must do to assure quality:

1. Find out who matters. Sort out *whose* ideas of quality matter, and what those ideas are.

2. The buck stops with you. You must accept responsibility for the quality of the *entire* product or service whose development or performance you manage. If you have a ship/no ship decision, you truly do own quality. You're the only person who can credibly claim, for the overall product, to *assure* quality.

3. To do that well, you must understand the work being done. Understanding comes with direct observation, participation, and interaction. If you observe by the numbers, instead of doing the work, you'll end up managing the numbers, not the work.

4. As a manager, it's your job to provide resources. (And by that I don't mean people. Please don't refer to people as "resources". Refer to them as people.) For your people, you must provide the tools, training, support, reference materials, and most of all, the time that your employees and contractors need.

5. And finally, you must remove any form of obstacle, measurement, or control that reduces people's intrinsic motivation on the one hand, and your ability to observe on the other.

So What Is Testing?

- "Questioning a product in order to evaluate it."
  - James Bach
- "Gathering information with the intention of informing a decision."
  - Jerry Weinberg
- "A technical, empirical investigation of a product, done on behalf of stakeholders, with the intention of revealing quality-related information of the kind that they seek."
  - Cem Kaner

Testing is NOT quality assurance!

With my co-author James Bach, I teach an approach to testing computer software called Rapid Software Testing, which we define as the fastest, least expensive testing that we can do while remaining credible, accountable, and focused on completely fulfilling the testing misison. In the course, we use the first of these three definitions.

Two things that these definitions have in common: they're all about questioning, investigation, an information search. The other thing is that none contains anything about assuring quality. Testing, and testers, don't do that. It's your response to the information and to all of the other information available to you—your awareness and your management actions—that assure quality. Testing helps with that awareness, but it isn't quality assurance.

What Is Testing?

Testing is the investigation of *systems* composed of people, products, and services.

- Excellent testing isn't *checking*; it's more like *anthropology*
- Anthropologists investigate many things
  - biology (human mechanisms; human "code" and "hardware")
  - archaeology (human history)
  - linguistics (human communication)
  - cultures (what it means to be human)

If you focus only on the product and its functions, you leave out questions of *value* and other relationships that include people.

Anthropology is highly multidisciplinary. While there are areas of focus, anthropology doesn't look at a single part of the human system; it looks at the elements of the system and the relationships between them.

One insight from anthropology, relevant to product management, is that other cultures—cultures that we sometimes refer to as "primitive"—are in fact quite sophisticated. They are specific adaptations to their context and their environment, and our own cultures would be maladapted to that environment. Cultures are always in flux, too.

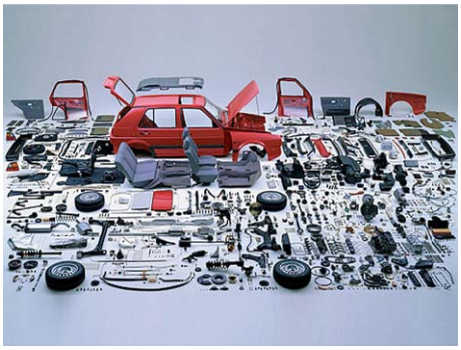
Wade Davis speaks particularly eloquently about this. See his books, including *The Wayfinders*, and see also his two TED talks:

http://www.ted.com/talks/lang/eng/wade_davis_on_endangered_cultures.html

http://www.ted.com/talks/lang/eng/wade_davis_on_the_worldwide_web_of_belief_and_ritual.html

Why We Must Test

- A product is **far more** than its components.
- Quality is **far more** than the absence of defects.
- Testing is **far more** than determining that some aspect of the product is “correct”.



Quality is value to some person(s).

Testing makes *informed* decisions about quality possible.

A product is **far more** than its components. For example, a software product is **far more** than the instructions for the device. A car is far more than the parts arranged in some formation; a house is not merely a set of building materials arranged in a house design pattern, as Cem Kaner puts it. A product is part of a system that also includes the services around it.

Quality is not merely the absence of defects. For example, a high-quality software product is **far more** than one with few errors in the code. You could have a program with no coding errors in it, a retail store with perfect inventory and information systems,

For example, serious software testing is **not about** writing test cases that return a pass or fail result. We can’t test for the absence of defects anyway. I can’t convince my daughter that there are no monsters under her bed. Even when we look, there can still be *invisible* monsters.

Testing is an *empirical* investigation: how will this product work in the world, and with the world?

Managers Are Like Testers

- A good tester doesn't simply follow scripts asking

Pass or Fail?

- A good tester *investigates* and asks

Is there a problem here?

In excellent investigation, self-management is key.
http://www.developsense.com/blog/2010/02/testing-and-management-parallels/

Excellent testers, like excellent managers, are good at self management.

Other blog entries on test management can be found at http://www.developsense.com/blog/category/management/.

Why Testing?

Excellent testing lights the way of your project.

- Testing is *applied critical thinking.*
- Testing consists of cycles of analysis, exploration, discovery, investigation, and learning.
- Whether you hire people called “tester” or not, testing is one of the things that makes quality assurance possible.

If you don’t hire testers, that’s okay. But someone had better be doing testing; that is, thinking critically about your product or service, and then applying that thinking to investigation of the nature of the product and the associated risks.

I like to think about risk in terms of three basic ideas:

1) A bad thing that could happen (like a programmer making a coding error, or a machine producing a misshapen widget).
2) The consequences of that bad thing (like the coding error permitting access to unauthorized users, or the installation of the misshapen widget triggering a product recall).
3) A chance that we’re willing to take. (We know that we’re uncertain about this area of the product, but we’re willing to risk shipping it without further testing.)

You can read more about risk here:
http://www.developsense.com/blog/2011/04/more-of-what-testers-find-part-ii/

So What Are We Testers?

Skilled investigators

> The tester doesn't have to reach conclusions or make recommendations about how the product *should* work. **Her task is to expose credible concerns to the stakeholders**.
>
> - Cem Kaner, *Approaches to Test Automation*, 2009 (my emphases)

Testers Are Sensory Instruments

The idea of testers themselves being test instruments comes to me from Cem Kaner. I've illustrated and extended the metaphor somewhat, but the basic idea is his.

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, but with a side effect: *information loss*.
Think of all 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*.

Decisions about quality are always political and emotional. Moreover, the problems that we must solve are often complex, and defy easy understanding. This can be scary. Many—even most—people would prefer at least to *appear* rational. Therefore numbers are often used to help simplify complex problems and justify subjective, emotional and political decisions.

That's okay, but there are risks.

The expression "partial answers that might be useful" is due to Cem Kaner, from his talk "Software Testing as a Social Science".

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, exactly? What's happening?"

People often quote Lord Kelvin: "I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of *science*, whatever the matter may be."[1] But, few note the sentence that precedes the passage: "In *physical* science the first essential step in the direction of learning any subject is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it." Note that he's not talking about quality in terms of value, but rather in terms of properties of things. (Oh, and by the way, Lord Kelvin also asserted that heavier-than-air flight was impossible, and that there was no practical purpose for radio.)

Kaner and Bond offer this definition of 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." I like this definition, to a point, but I'm still struggling with two problems. Kaner and Bond discuss and attack one of these problems quite thoroughly: the problem of construct validity. A *construct* is a class, category, or idealized (as opposed to concrete) object. Excellent measurement requires us to specify our constructs and the functions by which we apply numbers to them. (That's the difference between a measurement and a metric, by the way; a measurement is an observation linked to an evaluation. A metric is a function—a way of mapping a number to an observation. A construct is *valid* when the description and the classification map to reality in a reasonable way.

Problems ensue when constructs aren't valid. If the construct isn't valid, then neither the metric nor the measurement can be valid either. When that's a problem, some of your employees feel it—and the rest know it.

What Is Measurement?

Measurement is the art and science of making reliable observations.
—Jerry Weinberg

- 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".
- Measurement might be qualitative or quantitative, but assessment and evaluation are *always* qualitative:

What do we want?

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

See two articles on this: http://www.developsense.com/articles/2009-05-IssuesAboutMetricsAboutBugs.pdf, and http://www.developsense.com/articles/2009-07-ThreeKindsOfMeasurement.pdf. A related article, too: http://www.developsense.com/articles/2007-11-WhatCounts.pdf

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 are cheaper in every way.

Why Prefer First-Order Measures?

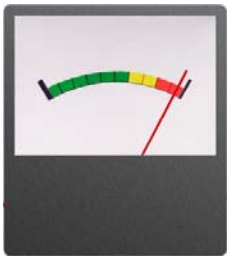
- When you're driving, are you mostly concerned about…
  - your velocity, acceleration, vehicle mass, drag co-efficient, frictional force? (third-order)
  - your engine temperature, RPMs, and current rate of gas consumption? (second-order)
  - looking out the window to avoid hitting something?

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

What kind of driver do you trust?

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.



This slide is taken from the work of my colleague, James Bach.
http://www.satisfice.com

A metric, by definition, is a simplification of reality, and as such, the same number can represent different realities.

Any system that involves humans is self-aware. *Any metric you use to control a self-aware system will be used by that system to control YOU.*

An inquiry metric might look like a control metric. The difference is how you use it and what you infer from it.

Observation can go directly to assessment and steering actions without quantified measurement. This is the first-order approach. Ask what other modes, beside numerical ones, you could use for observation and evaluation. Start by asking *what problem you want to solve* or *what situation you'd like to assess.* Prefer immediate observation to mediated observation. Make sure that you've considered a number of different possible interpretations, then choose a control action OR a search for more detail.

If you're worried that observations and assessments are subjective (they are), ask *several* people who matter

Obstacles

- In software testing (my field) 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/.

The Fundamental Regulator Paradox

- “The task of a regulator is to eliminate variation, but this variation is the ultimate source of information about the quality of its work. Therefore, the better job a regulator does, the less information it gets about how to improve.” —Jerry Weinberg

This quote is taken from *General Principles of System Design*, by Gerald M. and Daniela Weinberg.

Learning From Failure

- "Experience comes from bad judgment; good judgment comes from experience."
  – Barry lePatner
- "Try again; fail again; fail better."
  – Samuel Beckett
- Success comes from learning, but learning comes mostly from failure
  - see Henry Petroski, Charles Perrow, Dietrich Dörner, and others on engineering
  - see Gever Tulley on education http://www.youtube.com/watch?v=hvHViFc0ekw

Gever Tulley's video is deeply inspiring for anyone interested in learning. TED Talks (http://www.ted.org) contain a wealth of wonderful learning-oriented presentations.

Learning from Failure

- In our teaching and practice of software testing, James Bach and I advocate accelerating learning by combining
  - increasing freedom and increasing responsibility
  - close personal supervision
  - continuous feedback and increasing accountability, via…
  - conversation reinforced by *concise* documentation
  - in a fault-tolerant environment.

This is of the essence of our approach towards exploratory testing, too. It's designed to reinforce rapid learning about a product, and rapid development of skill. This means that, in short order, testers need not depend upon personal supervision, formulaic procedures, or favourable environments to do excellent work.

You can read about our classes at http://www.developsense.com/courses.html. You can also peruse my Web site and James Bach's, http://www.satisfice.com for more on what we have to say about testing and tester skill.

Positive Deviance

- In any population, there's one person who is outperforming all the others. What does that person do?
- One catch is that you have to observe the work as it's being done, which means observing a system that includes
  - individuals
  - related individuals and groups
  - artifacts
  - outputs
  - resources

See *The Power of Positive Deviance: How Unlikely Innovators Solve the World's Toughest Problems* (Pascale, Sternin, and Sternin)

See also "The Palmer Method" (video), http://www.positivedeviance.org/resources/flv/jasper.flv

"Positive Deviance is based on the observation that in every community there are certain individuals or groups whose uncommon behaviors and strategies enable them to find better solutions to problems than their peers, while having access to the same resources and facing similar or worse challenges.

"The Positive Deviance approach is an asset-based, problem-solving, and community-driven approach that enables the community to discover these successful behaviors and strategies and develop a plan of action to promote their adoption by all concerned."

http://www.positivedeviance.org

The Palmer Method video is a demonstration by a patient escort, Jasper Palmer, who demonstrates a method for removing a contaminated paper gown after it has been in contact with a potentially contaminated patient. The problem was that gowns would quickly overflow garbage containers, exposing staff and patients alike to methicillin-resistant staphylococcus aureus (MRSA, commonly known as a superbug). Watch how this fellow solves the problem!

Leadership

- “Creating an environment in which everyone is empowered.”
- Not all managers are leaders, and not all leaders are managers.

In my community, leadership means "creating an environment in which everyone is empowered". In groups of people, there are many people who lead by example, by honouring the contributions of others, by speaking up when it's time to speak up, by teaching or mentoring or coaching, by allowing themselves to be taught or mentored or coached. Such people don't always carry the title of manager. Often they're regular employees, individual contributors. Sometimes we speak of "thought leaders" in a group. We might also consider "community leaders" who never seek election to public office, but who participate at the food banks, write letters to the editors and the politicians, help to organize and staff community events. We might not think of those people as leaders, since they don't have titles or elected offices or supervisory responsibility, but they are leaders nonetheless. They make it possible for other people to contribute and to have better lives. That's the kind of leadership I'm speaking of here.

Conversely (and sadly), some people with the title "manager" do few or none of the empowering things I've noted above. The great ones do all of them, and I've had the privilege of working for, and with, a couple of those.

For more on that topic, see /Becoming a Technical Leader/, by Gerald M. Weinberg. Also see /The Power of Positive Deviance/, by Pascale, Sternin, and Sternin.

Who Contributes to Quality?

- Designers and builders *deliver* quality by creating and producing things of value.
- Testers help to *defend* quality by exploring, discovering, and investigating problems that threaten quality.
- Managers *assure* quality by empowering people, providing resources and removing obstacles.

Michael Bolton

http://www.developsense.com
michael@developsense.com

References: Jerry Weinberg

- Perfect Software and Other Illusions About Testing
- Quality Software Management
  - Volume 1: Systems Thinking
  - Volume 2: First Order Measurement
- Quality Software Management: Requirements Before Design
- An Introduction to General Systems Thinking
- The Psychology of Computer Programming

– Jerry Weinberg

References: Cem Kaner

- The Ongoing Revolution in Software Testing
  - http://www.kaner.com/pdfs/TheOngoingRevolution.pdf
- Software Testing as a Social Science
  - http://www.kaner.com/pdfs/KanerSocialScienceSTEP.pdf
- Software Engineering Metrics: What Do They Measure and How Do We Know? (with Walter P. Bond)
  - www.kaner.com/pdfs/metrics2004.pdf
- Approaches to Test Automation
  - http://www.kaner.com/pdfs/kanerRIM2009.pdf
- Lessons Learned in Software Testing
  - Kaner, Bach, & Pettichord

Book References

- The Black Swan
- Fooled by Randomness
  - Nassim Nicholas Taleb
- Secrets of a Buccaneer Scholar
  - James Bach
- The Power of Positive Deviance
  - Pascale, Sternin, and Sternin
- Sciences of the Artificial
  - Herbert Simon
- How Doctors Think
  - Jerome Groopman

Book References

- Measuring and Managing Performance in Organizations
  – Robert D. Austin
- Tools of Critical Thinking
  – David Levy
- Mistakes Were Made (But Not By Me)
  – Carol Tavris and Eliot Aronson
- How to Lie With Statistics
  – Darrell Huff
- The Visual Display of Quantitative Information
- Envisioning Information
- Visual Explanations
- Beautiful Evidence
  – Edward Tufte

Who Does Quality Assurance?

April 29, 2011
Follow-up Questions to the Webinar

Q. How do you ensure that you are focussing on quality assurance and adding value and not being perceived as policing or auditing?

If you're a tester, I think the easiest way to deal with that issue is to stay out of the quality assurance business entirely (except to assure the quality of your own work). As testers, the value that we add is in the form of information about the product and the project. We don't produce or change or manage the product, but we add to the system of the product and what people know about it. In that sense, we're like investigative reporters, not police.

Q: I would assert "Checking" is still very important and testing would not even matter if you didn't first check. Is it (or should it be) the same person responsible for checking and testing? Should testers own both?

A: Yes, checking is very important. (I've written a lot about checking; see here http://www.developsense.com/blog/2009/08/testing-vs-checking.html and here http://www.developsense.com/blog/2009/11/merely-checking-or-merely-testing/ for entry points.) At the unit level, checks provide the first line of defense against problems that might otherwise be buried and much harder to find later on. I would argue that checking tends to be cheapest and easiest when programmers do it, with the added benefit that the feedback loop is very short. There's a risk there, though: a programmer's checks will often recapitulate the problems that are in the production code. That's why pairing and other forms of review are important too. Jonathan Kohl has noted that testers' ideas about risk are different from programmers' ideas about risk, and he has written some very useful stuff on why that can be both a good and a bad thing in various contexts. Checking and testing both benefit from collaboration and review, it seems to me.

It's not that testing wouldn't matter if you didn't check first, but it would be harder and slower. In particular, there would be more noise in the process. I've written a good deal about that here: http://www.developsense.com/blog/2009/11/why-is-testing-taking-so-long-part-1/

I'm more skeptical about the cost and the value of checking higher up. At the lower integration levels, it seems to remain relatively cheap and easy for a lot of purposes. *Very* generally, it seems to me that the higher up you go, (programmed) checking is harder to do, takes longer, and is less valuable considering that lower-level checks are more likely to have found the kinds of problems that yield to checking. In addition, an emphasis on checking presents opportunity cost against other kinds of testing we could do, and it might (*might*) tempt us to relax our vigilance for problems that we didn't anticipate. So I'd emphasize more human interaction the closer we get to human interfaces. If the program lends itself well to rapid development of checks at higher levels, some checking might be appropriate. Other people are strong advocates for that. My own experience is that it's

easy to get sucked into programming high-level checks when I might better spend my time looking for new problems and new risks. That's why diversity—lots of different people, models, approaches, preferences, temperaments—on a team is really important.

One more note: checking *requires* tools, but just about any kind of testing can be *supported* by tools. I'd like testers to think very expansively in terms of how tools might help them do all kinds of things—visualization, data generation, recording, reporting. So checking is not the only application for tools, by any means.

Q: At the end of your presentation you made the point that "leaders" are sometimes not "managers" — please expand if possible.

A: Sure. In my community, leadership means "creating an environment in which everyone is empowered". In groups of people, there are many people who lead by example, by honouring the contributions of others, by speaking up when it's time to speak up, by teaching or mentoring or coaching, by allowing themselves to be taught or mentored or coached. Such people don't always carry the title of manager. Often they're regular employees, individual contributors. Sometimes we speak of "thought leaders" in a group. We might also consider "community leaders" who never seek election to public office, but who participate at the food banks, write letters to the editors and the politicians, help to organize and staff community events. We might not think of those people as leaders, since they don't have titles or elected offices or supervisory responsibility, but they are leaders nonetheless. They make it possible for other people to contribute and to have better lives. That's the kind of leadership I'm speaking of here.

Conversely (and sadly), some people with the title "manager" do few or none of the empowering things I've noted above. The great ones do all of them, and I've had the privilege of working for, and with, a couple of those.

For more on that topic, see *Becoming a Technical Leader*, by Gerald M. Weinberg. Also see *The Power of Positive Deviance*, by Pascale, Sternin, and Sternin.

Q: What do you do when micro-managing threatens quality?
A: It depends who the "you" in the question is—the person micromanaging, or the person being micromanaged. Micromanagement, in my observation, represents the collapse of trust. So for managers, one of the first questions to ask is "why don't I trust my people?" For an employee, a key question is "Why doesn't my manager trust me?" Often the problem is that the tester isn't as good as he or she might be at describing or accounting for his or her work. In testing, that depends on a skill that James Bach and I call test framing. (You can read about that here: http://www.developsense.com/blog/2010/09/test-framing/). In other lines of work, it might be worth looking at similar patterns for making the work credible and accountable.

Presumably someone is micromanaging because they're concerned about a particular risk. So there are also risk questions—what could go wrong? What harm would it cause? Are there ways, other than micro-management, to mitigate the risk or the harm?

It seems to me that there are several threats to the quality of any product or service when managers supervise too closely. A micromanager, like everyone else, has a limited set of models for the work and the product. When you control things such that there's only one way of doing things, that limits the ability to find new problems, new perspectives, new experiments, novel solutions. A manager who is total enmeshed in his employees' work is probably dedicating less time than he ought to providing resources and clearing obstacles.

A person being micromanaged becomes solely an extension of the manager's capabilities, and brings little of his or her own value to the table, yet there is surely value there to be had. Worse, that person doesn't get to learn from experience. It's really important for people to be able to try things and fail. Ideally those failures happen in ways that aren't consequential, so a good manager sets up ways to make the project robust to larger failures while permitting and even encouraging little failures. Failures are where information and learning come from. A good manager allows his or her people to blossom, but that means granting them freedom and responsibility and a little risk.

Measuring and Managing Performance in Organizations by Robert D. Austin provides a fabulous and rigourously argued discussion of why micro-management is a bad idea.

Q: IMO, the presenter is trying to use testing to counterweigh architecture/design mistakes, a compensating control, which is hardly an adequate respose.

A: I appreciate the idealism that seems to be at the core of this comment. I'd love to be confident that mistakes never happened, and I'm aware that there are certainly many things that we could do to reduce or avoid mistakes. However, the idea that testing is a compensating control is, in my view, a misperception of what testing is, and does. To me, testing itself isn't a control at all. Instead, it's a questioning process, designed to identify whether there are problems. That's something that's worthwhile, even after we've applied what we think are our very best efforts. To put it another way, testing isn't a compensating control; it's a way of determining whether—and where, perhaps—we need some kind of control, compensating or otherwise.

Testing is the means by which we seek problems in the development work that's just been done. In that sense, testing is as much a part of development work as programming is. The search for problems can happen after a huge amount of development work, or after each atomic unit of work. The Agile development people have some good ideas, I think, in suggesting that we develop and test in very tight loops, right at the unit level. Many suggest that we work most quickly by devising one or more tests before we develop the code that is intended to pass those tests. Then we develop the code such that it passes the test(s), and we also check to see that the code continues to pass all the tests that have passed before. The wisest Agile people that I know also recommend that we explore the product after that, looking for problems that our earlier tests and checks have missed. I agree with that approach.

In other, more traditional models of development, I observe that sometimes people confuse testing with fixing. Companies often break work into a long "development phase" and a long "testing phase". I don't agree with this separation, especially since the "testing phase" lumps testing and *fixing* together. In the "testing phase", the testing takes relatively little time; it's the *fixing* that takes the majority of the time. So, perhaps the statement above refers to the idea of *fixing* as a compensating control on architecture or design mistakes. That seems wasteful to me too, although as I've said above, we do learn mostly from failures. If there were more testing—by which I mean more review, more thinking critically about the product—in the earlier stages, then perhaps we wouldn't have to deal with so many problems later. So if that's what the comment is leaning towards, I agree.

Q: QA is a subfunction of the governance. It's about doing the right thing, where QC (aka testing) is doing it right. As such, the subject of the presentation is somewhat misleading, in my very subjective opinion. Any comment?

A: You and I appear to have different ideas about quality assurance and quality control. I'm aware that some people see quality assurance as a form of governance. I suppose one could think that way. I prefer to think of quality assurance in terms of two forms of management—self-management on the part of the individual, as he or she does his or her work; and organizational or project-level management, as done by people who manage the work overall. Perhaps the latter a kind of "governance" function, but I prefer to think of it more simply as *management*, if it comes with actual management authority. (If it doesn't come with actual management authority, then the "quality assurance" moniker and role make me feel uncomfortable. On the cop shows, no one likes the Internal Affairs department.)

I think of testing neither as quality assurance nor as quality control, since to me testing is a questioning process, an information-gathering process. The search for information can neither assure nor control quality. The search and the discovery can only inform. It's what people think and what people do with the information that assures or controls quality.

So: misleading? Perhaps, if you enter the discussion with a particular model, and if we don't share that model. I'm sorry if you felt misled. On the other hand, we could choose to think of our differences as potential viable alternatives, and we could share those, maybe coming up with something better than either one alone. At least one of us will learn something!

Q: Generally we find that the people doing user acceptance testing are the business analysts who do the requirements definition. Any comment on roles & responsibilities?

A: If you're going to ask people to do anything, I think it behooves you to help them get good at it. So when business analysts are asked to do testing, I think it behooves the people asking to provide plenty of support for doing testing really well. That might

include bringing in an expert tester to work alongside them, providing training, providing books, sending people to classes or conferences. Airlines can't put a doctor on every plane, so flight attendants end up having to do first aid and CPR sometimes. But airlines don't want flight attendants to do CPR incompetently. It's important work, so they train them and retrain them.

There's nothing wrong with BA's doing testing, but there's a good reason to have more people doing it. People tend not to be good at seeing problems in their own work, so a BA's testing might mirror any problems with the BA's initial analysis. So James Bach, Cem Kaner, and I (to name but a few) argue very strongly in terms of diversity on a testing team, in all kinds of dimensions: speciality, educational background, domain experience, programming knowledge, other technical skills, critical thinking, systems thinking... People from different cultures and ethnic backgrounds see things differently. If we're looking for problems in globally distributed products—in *any* products—I think that's a good thing.

Q: Can you give some examples of inquiry vs decision metrics?
A: In general, an inquiry metric is one that is designed and used to prompt questions. A control metric is one on which we base a decision or an evaluation. I gave an example towards the end of the presentation. There are two ways that you might collect data on the number of calls handled by technical support people.

If you set a goal for each person to handle 28 calls a day, they will feel pressured to do that, even if they don't admit it, and even if they don't do it consciously. Some will infer that more is better. Some who are capable of handling calls quickly might ease up, managing themselves down to the number.

If you decide instead to collect data to prompt questions, there will be a different outcome. The first thing is that you'll probably observe outliers, one way or the other. Does the person who handles a large number of calls have techniques and approaches from which other people might benefit? Does the person who is handling few calls do so because she has an affinity for tough problems, and happily takes them on for the other team members, allowing them to get higher call counts? Or is she lagging, in need of training? Take the measurement in, and subject it to a number of possible interpretations, and you'll be less likely to miss important information.

You can choose to put control metrics on a bunch of things at once, but that results in a kind of quality whack-a-mole: as you add measurements, the things that you're not measuring but that are nonetheless valuable will suffer. Again, *Measuring and Managing Performance in Organizations* by Robert D. Austin analyzes this problem really thoroughly, and suggests that intrinsic motivation and a basket of qualitative goals are key. Also see Jerry Weinberg's *Quality Software Management* series, the whole thing, but for this issue, especially look at *Volume II, First Order Measurement*. (Jerry's books are available as e-books through his Web site, http://www.geraldmweinberg.com. In that case, the book I've just cited is distributed as two e-books: *How to Observe Software Systems* and *Responding to Significant Software Events*.

Q: There seems to be a digital thought to Testing or QA. Especially in software, products/services are continuous - should not testing and QA been thought of more as continous/lifecycle activity?

A: I think so. Binary thinking is pretty limiting. I like the analogy to discrete and continuous math! The way I see the world, none of software, or testing, or quality is decided by bits, but by continuous waves of different quality criteria sloshing around in a big, noisy ocean.