The Rise of the Intellectual Indian Software Tester

Part 1

- by James Bach

To be a tester is to be a thinker. Testing is not just pressing keys and looking at the screen. Testing is pondering and learning and interacting with a product for the purpose of discovering important and hidden truths about it. Testers are troublefinders.

Strictly speaking, this process cannot be scripted. A scripted process is one that is determined in advance of its execution. One drawback of a fully scripted test process is that it can only find problems that were *specifically and precisely anticipated* at the time the script was produced. But when we test we need to find unanticipated problems, too. We want to find *all* the problems that matter. Another reason testing cannot be scripted is that human curiosity, learning, and confusion—all of which are crucial to a test process—cannot be reduced to an algorithm. Complex problem solving, in general, involves substantial tacit as well as explicit skill.

This brings me to my topic: the Indian testing industry. It is commonly believed in the West that Indian testers cannot work without scripts; that they cannot think in rich and deep ways while testing. Meanwhile, in the last twenty years, in the West, a small but passionate and growing community of testers has been engaged in an intellectual testing. (This is not the "agile testing" movement. I'm referring to the community that calls itself the Context-Driven School. We focus on skills that allow us to operate effectively in any context.)

Intellectual testers don't need notes pinned to their sleeves in order to take action. We design and re-design our own tests, as we go. We are systems thinkers, creative thinkers, and critical thinkers. This is not exactly new, but we brought a new element to it: *systematic* and *collaborative* skill development.

For the first years of this revolution, India played no role in the Context-Driven movement. It was a dark continent. That has now changed.

My First Visit

My introduction to India left me with mixed feelings. I visited Bangalore the first time, nine years ago, at the behest of two of my clients. Both large American companies, they had outsourced some of their testing, but were disappointed with the result. The way my clients described their Indian vendors: they weren't testers at all. They were, at best, script jockeys. This is what my community calls "factory school" testing. (My community calls itself the Context-Driven School. We focus on skills that allow us to operate effectively in any context.)

Check around the Internet. Talk to test managers. This is a common story. You'll hear it all over: "Upper management forced us to outsource to India, and *those testers suck*. (eye roll and sigh.) Oh well, there's nothing to be done."

And so I went to India. But what I found surprised me. This is what I wrote in 2003, just after I returned:

> *I have met quite a few such sharp testers in the U.S. and UK. But, in India I expected to find polite, silent students. I expected that they would be intelligent, but timid about engaging mind-to-mind with me in class, especially when what I'm teaching flies in the face of most traditional advice about testing that they have read and heard.*
>
> *What I found instead were testers who quickly warmed to the challenges I made. They did speak up. They didn't challenge me with the same intellectual swagger typical of testers in, say, England, but they responded to my questions and found novel answers to problems I posed them. In several of the exercises, solutions were proposed that I had not heard from anyone else since I started teaching in 1995.*
>
> *Like most testers, they have not yet developed their potential. But I no longer believe there are important cultural obstacles to hold them back. So, don't be surprised if in a few years the Indians start getting the reputation as insightful, penetrating testers.*

I admitted I was wrong about Indian testers. My brother Jon discovered the same thing and blogged about it six years later, when he trained an Indian team of his own.

> *They loved exploring, were not shy, talked over each other, even gaggled like kindergarteners eager to show each other as if it was show-and-tell time. It was amazing, and it was as easy as a key being turned in a lock.*

Indian Testers Are Not Stupid… So Why Do They Test Badly?

Imagine someone who owns a sports car with a 350-horsepower engine, and yet pulls it around with an elephant—not because gasoline is expensive, but because he thinks turning on the engine would seem arrogant, presumptuous, and maybe the sound of the motor could disturb the neighborhood.

That's my feeling about Indian testers. I met lots of smart people in my Bangalore testing classes. But I got the impression *they didn't feel it was polite to think like a tester*.

Indian culture is not an anti-intellectual culture. But it is a somewhat hierarchical, family-oriented, *service* culture. Good Indians listen to their parents and do their duty as they see it. This attitude also affects their sense of what is polite to say to the boss (or parent or foreign client as the case may be). They want to say "yes" even when they feel the honest answer would be "no" or "I don't understand." This can give the impression of evasiveness in a technical project. Good testers say unpopular, important truths. Testing is a service occupation, yet a misguided sense of service—not wanting to annoy the boss—can ruin the process.

That's one reason for the trouble. Here are some others:

1. English is not the first language of many people in India, making communication inherently difficult in an industry dominated by English speakers.

2. Outsourcing is inherently difficult, but since India is such a popular outsourcing destination, the problems of outsourcing become unfairly confused with the problem of being Indian.

3. A culture that routinely employs another culture is likely to begin thinking of itself as more "senior." It is in a position of power and experience, regardless of its actual competence. Therefore, Western companies may be *predisposed* to thinking of their foreign vendors as confused and timid.

4. No one ever taught them how to test. This is not specific to India, of course, but when combined with cognitive biases known as *Actor-Observer Asymmetry* and *Trait Ascription Bias* (I'm sure I don't need to explain these, because you are good at Google, too) it creates the impression that those guys over there, who happen to be Indian and who seem to be incompetent, are *more* incompetent than our own people (even though they aren't) and are incompetent *because* they are Indian (even if Indian heritage has nothing to do with it).

Taken together, I actually felt this was good news for India. Because it's *fixable.*

> Yes, yes, it seemed clear to me that Indian testers, in general, were pretty bad—but that's just like testers everywhere else. And yes Indian culture looked like a mild handicap, and certainly test outsourcing was particularly hard to do well under the best of circumstances.

But even so, I saw a wonderful potential there, once testers in India began to wake themselves up. I figured it would take a few years.

The Coming of Pradeep

Indeed, it took two years, as I reckon, before the transformation began to happen. It came into my life in the form of a particular man.

January 10th, 2006

Dear James,

Surprising why a stranger addressing you as 'dear', well the reason being we both share something in common - Testing. I introduce myself as Pradeep Soundararajan from Bangalore, India who happened to look through www.satisfice.com.

With that quirky intro, Pradeep requested to become my student. I like having students, but my time is limited, so I gave him an assignment that would require him to do a lot of work: I asked him to test a web site, expecting not to hear from him again. Instead, he completed it the next day.

At that time, I was in the middle of a very intense court case, with very little time to spare. It would take Pradeep 98 days, sending me 19 reminders, before I got around analyzing his work. When I did, I was pleased with I found, and very impressed with his determination. I decided to invest in him.

I don't necessarily charge money to teach testing. Instead, I provide coaching and support to people who have the drive to keep sending me reminder emails. Also, I help people who inspire me, and one thing that inspires me is when they turn around and help others. Pradeep became one of those students; he was a catalyst for the new wave of testing enthusiasts, centered in Bangalore.

Very soon after he contacted me he started a blog, partly to share his testing ideas and partly to practice writing in English. His blog was a call to Indian testers to wake up, and some testers heard his call and responded. He held meetings and taught classes. A new, small, Indian testing community was born. Now it seems like there are a lot of Indian tester blogs devoted to fostering testing skills, but Pradeep was the first of those. He deserves a lot of credit for that, and many of the bloggers he inspired have now come to me for coaching, too.

Another factor that helped was the arrival of the Black Box Software Testing online class. Created by Cem Kaner, with the support of some of my materials, it is a demanding multi-week testing learning experience. People all over the world have signed up for it, including many in India. Between Pradeep's preaching, online resources, remote coaching by Context-Driven testing guys like me, and the BBST class, any ambitious tester in India can now get the support he needs to thrive.

India as a Testing Cuisine

My exposure to Pradeep and others such as Ajay Balamurugadas, Meeta Prakash, Shrini Kulkarni, and Parimala Hariprasad, began to change and broaden my perception of Indian culture and its potential role in building great testers. I began to see the possibilities of Indian testing against the backdrop of thousands of years of philosophy and history.

Please bear in mind that my excitement about India is not a comment on any other culture. Just as I like to eat at Japanese and Italian restaurants, I also enjoy Indian food very much. This is not to say that there's anything wrong with traditional American cooking. In the same way, although I am an American tester, and I appreciate the advantages that American culture brings to the test process, I also am excited about how I can be a better tester by learning about India.

I have dabbled in the study of India for years. Only recently have my studies become focused. I'm interested in the relationship between "thinking like an Indian" and "thinking like a tester." I'd love to see a few Indian testing heroes stand up and do this study better than I ever could, but to get you started, here are some of the elements that can inform excellence of Indian testing:

- **India is a plural society that knows how to adapt.**

  Where else can we find so many people, with so many different religions and sects, living so close together with so little violence? Nowhere. Growing up amidst such diversity may help Indians in technical life, too. Testing requires us to bring together competing ideas and interests.

- **India is especially patient in the face of chaos.**

  To a Western eye, Indian cities are an utter mess; a tsunami of dysfunction. But somehow they seem to work, and even thrive. What sort of mind must be required to tolerate living there? Imagine that same mind encountering chaos in a test project. It may be easier for Indian testers to remain calm through the long parade of outrageous bugs.

- **India understands service, loyalty, and reverence.**

  India is a family and clan-oriented culture. They are generally more comfortable with hierarchy than we are in the west. Whereas this may be a drawback for independent creative thinking, it may a boon when it comes to daily motivation and reliability—as long as the needs and tasks of testing can be framed in terms of service.

- **India has its own tradition of Epistemology.**

  Epistemology is the branch of philosophy concerned with how we know what we think we know. One of the great concerns in Hindu and Buddhist tradition is distinguishing reality from illusion. So, it should not surprise us that India has an ancient tradition of logical, skeptical, and scientific thought that actually pre-dates the Greeks. Most Indians don't study it and don't even know about it, but it's there as a source strength for those who do, because testing is nothing more than applied Epistemology.

- **India has a rich tradition of heuristic learning.**

  Treasures of Indian literature include the Mahabharata (and Bhagavad Gita), Thirukural, Arthashastra, Panchatantra, and the Buddhist and Nyaya Sutras. Something all of these have in common is the practice of teaching through the consideration of opposing ideas and outright paradoxes. This fosters what I call heuristic learning, which develops the ability to make complex judgments where no clear right answer exists. Testing requires that sort of thinking, because of the impossibility of complete testing.

- **India understands that excellence is achieved through struggle.**

  One of the differences between India and America is that here in America the culture expects instant gratification. Both in the everyday social order and the spiritual literature of India, however, great outcomes are expected to take time and work. This is important because it requires a long, daily struggle, and much experience, to develop deep testing skills.

I would like to thank Michael Bolton and Mary Alton for their help with art and editing. In part 2 of this article, I will describe my return to India after nine years. I taught at Intel, and Barclays, re-visited Mindtree, and spent several days with Pradeep and the crew at Moolya. The star of Indian testing is rising.

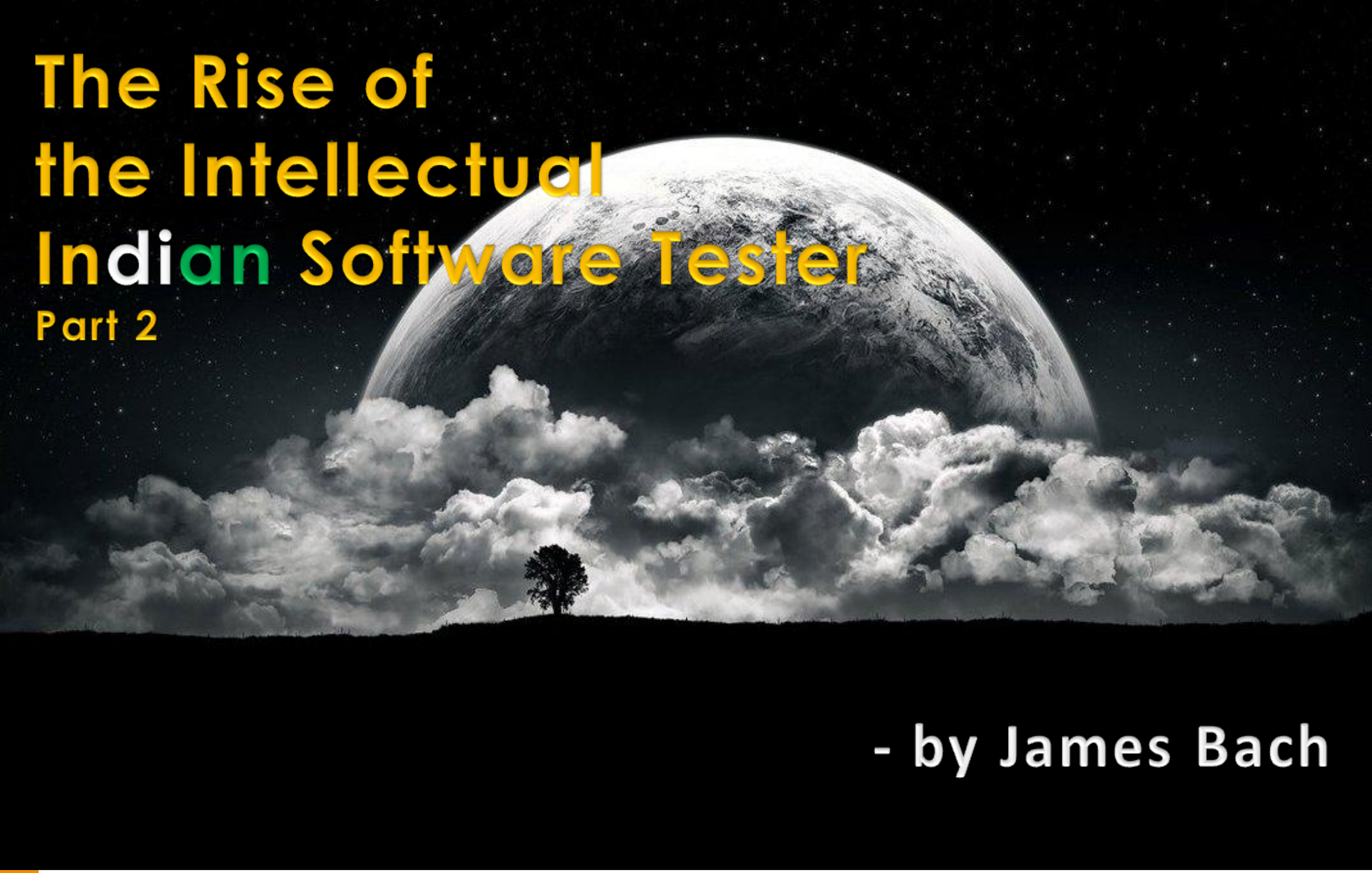
James Marcus Bach is a software tester, author, trainer and consultant. He is a proponent of Exploratory testing and the Context-Driven School of software testing, and is credited with developing Session-based testing.

His book "**Lessons Learned in Software Testing**" has been cited over 130 times according to Google Scholar, and several of his articles have been cited dozens of times including his work on heuristics for testing and on the Capability Maturity Model. He wrote numerous articles for IEEE Computer.

Since 1999, he works as independent consultant out of Eastsound, Washington.

He is an advisor to the Lifeboat Foundation as a computing expert.

Follow James on Twitter @jamesmarcusbach or know more about his work on satisfice.com



The Rise of the Intellectual Indian Software Tester

Part 2

- by James Bach

[Please read part 1 of this series in *December 2012* issue of Tea-time with Testers]

So, I returned to India.

People kept asking me what I thought of all the changes to Bangalore since my last visit almost a decade ago. "It looks exactly the same to me," I replied, which says much more about my first visit than it does about the city itself.

The first time, I didn't *know* anyone, and Bangalore is not a welcoming city to tourists from the West. I was stuck in my opulent hotel, able to see some palm trees through my window and a few hazy buildings in the distance, strung with clotheslines. Through the windows of the taxi each day I witnessed families piled onto scooters, burning trash, wandering cows and wondered what was the life expectancy of a Bangalore cabbie. All this muddled together into a blurry soup in my memory. And I was never really *in* the soup. Shuttled from hotel to worksite, never setting foot in the country itself, I didn't visit India so much as I *interfaced* with it. No wonder that when I returned, it was as if to a place I had never been.

This time I *did* visit. Today I know a couple hundred testers in India, and many more know about me. That gave me the opportunity to get out of the hotel properly, and meet people, and dine at their homes.

The growing community of intellectual testers in India is my tribe. Regardless of nationality or language or politics, I feel kinship with anyone who works to build his mind into a better instrument of analysis and observation.

Though still greatly outnumbered by "factory-style" testers, all around the world this community is growing. India is probably the highest growth area, today, for intellectual software testing. I have no hard statistics to back that up. It's just an unscientific impression— but I've never been more mobbed by supporters than I was on this trip. I felt like a minor Cricket star, at times.

I got to see a fascinating cross-section of companies on this journey:

- **Moolya,** a brand new test lab, completely free to innovate.
- **Barclays Bank**, a large company that has made a strong commitment and investment in skilled testing culture.
- **Intel**, a large company mired in traditional testing, but starting to come around.
- **Two established outsourcing companies** considering how (and whether) to explore the potential of a skilled testing culture.

James inspecting testing at Moolya

What is "traditional testing?"

People often contrast what I do with "traditional" testing, as if it is older than my own tradition of context-driven testing. But they can't tell you where their tradition comes from. I'll tell you: it's from 1972. That was the year the first testing book was written, by Bill Hetzel. That book, called *Program Test Methods*, documents the proceedings of a conference on testing held at Chapel Hill, North Carolina. It's not the first writing about testing, merely the first entire book. And it's the first time (based on my review of journal articles, IFIPS conference proceedings, and other writings from the sixties) that the complete "Factory School" and "Analytical School" visions of testing are asserted all in one place. For example:

- Document everything in a specification, then test strictly to that specification.
- Encapsulate all testing in units called "test cases" which can be counted. Test cases either pass or fail.
- The role of testing is to "ensure quality" or "certify the product" or "prove correctness."
- Test coverage means code coverage.
- All testing should be automated, if possible.
- Testing is embodied in specific definable actions or equations, which can be optimized with rigorous mathematical or manufacturing methods.

What is "skilled testing?"

What's left almost completely unwritten in *Program Test Methods*? Anything about skill. That is, the book is nearly silent about how a person goes about becoming a good tester, or what a good tester must know how to do. Humans are treated as wet, squishy afterthoughts, rather than the designing and controlling intelligence that makes testing go.

The skilled testing revolution turns that around. We look squarely at people, what people can do, and how people learn to do that. Tools, artifacts, and measurements all follow and serve people.

The first implication of skilled testing is that testing skills can and should be *systematically developed*, rather than assumed to come into existence through a simple act of will or divine providence. This requires *deliberate practice*, and an effective *mental model of testing*. When I sat down to create my own mental model of testing, I found that the fields of *Epistemology*, *General Systems Thinking*, and *Cognitive Science* provided the raw material I needed to develop it. Then I had to learn about *heuristics* and *bias* and the nature of *tacit and explicit knowledge*.

Developing deep skill as a tester requires letting go of cherished illusions such as the belief in specifications that define complete and correct expected results, or tools that test while you sleep, or test cases that have only two outcomes: pass or fail. The skilled tester accepts a world brimming with ambiguity but still intelligible to the determined and subtle mind.

To a casual observer, none of this is obvious. It may even seem like I'm over-complicating it. Only when a tester makes a decision and a commitment to becoming an excellent tester can he penetrate the surface layers of dead leaves and broken buzzphrases and begin to see the majesty of our chosen craft. For a large company to do this, strong leadership and a supportive culture is required.

Moolya, Young and Hungry

I'm droolin' over Moolya[1].

I would not have returned were it not for the Moolya test lab. They sponsored my trip. Intel and Barclays Bank later called "me too!" but my friend and student Pradeep Soundararajan got things going. My wife and I were a bit confused at first, because we didn't think Moolya could afford my fee. We had the impression that Moolya was just a few testers in a garage. Actually they've grown a lot in two years, moved out of the garage and into a nice modern building, with about fifty testers now and poised to double its business this year.

I arrived there to make an inspection tour. I wanted to discover if Pradeep[2] was really running a company for thinking testers. The quick answer is yes. Really yes. He definitely is.

I was touched to see all work stop when I walked in the door. (I hope no client had an impending release due). Suddenly I was surrounded by smiling, eager faces. *Young* faces. Moolya discovered (as I also have) that it's generally easier to train novices to be skilled rapid testers than to hire experienced conventional testers and try to convert them. It does take a long time to master testing, but at least with novices you don't argue as much, don't need to pay as much, and they are probably more loyal.
Having said that, an experienced intellectual tester is a godsend, and any good test lab needs them. Pradeep was wise to hire Dhanasekar Subramanian (DS for short) and Parimala Hariprasad, early on. Both of them had distinguished themselves as introspective and insightful testing bloggers.

After a few minutes giving what I hoped was an inspirational speech, I got down to the inspection. First stop: mobile lab. DS is "commander" of the mobile testing lab at Moolya, but he stood aside like a proud father and let me interview some of his team.

First thing they showed me was a series of mind maps. They claimed that they managed all the testing with mind maps rather than traditional scripted test cases. I can believe it. I saw a lot of mind maps! Nested ones, big ones, colorful ones, icon-encrusted ones. Test ideas are stored there as well as test notes and test reports. D.S. and his guys have developed mind mapping in testing to a new height of sophistication.

I was particularly intrigued by a mind map they designed to coordinate the actions of five testers during a one hour testing blitz. A branch of the tree was dedicated to the activities of each tester, and color-coding was used to divide the hour into segments. It was a beautiful, clean, organized plan, all on one page. I hope they publish it, someday.

Anyone can make pictures. I wanted to see if they knew what the pictures meant. I picked a fellow standing nearby, pointed to a leaf node on the mind map, and asked him what specifically he did to test when he read that. He described a fairly vivid process, but I thought I may have accidentally asked him

---

[1] Bear I mind that I have a vested interest in Moolya. The company was co-founded by a student of mine and they sponsored my trip. I *think* I'm being honest, here, but it's possible that Pradeep's wife's cooking was good enough to disturb my objectivity. Be on your guard.

[2] I say Pradeep runs the lab, because he is in charge of most of the operations and owns the culture, as such. But co-founder Santhosh and CEO Mohan play vital roles, too.

about something he was an expert on, so I picked a girl standing next to him and pointed to a different part of the map. She also gave a good answer.

I listened for hesitations. I listened for buzzwords and pablum phrases. What I heard thrilled me, not just the content but the tone of it: eager to impress, eager to respond.

How to do a spot review of a test process

It's not about documents, although documents will give you clues. It's not about what people say, although that can help. It's really about what the testers are doing: testing is what testers do. To review a test process, therefore, you need to collect observations about testing in action, see the traces of testing past, and draw inferences from that like a detective. It really is like being a detective, because you must be alert for various ways people can hide their process, either on purpose because they are ashamed of it and want to tell a better story, or accidentally because they don't know what their own process is or don't have the skill to express it in words.

Here's the approximate procedure:

1. **Begin by having done many inspections of test process before.** I know that sounds strange. If this is your first time seriously analyzing a test process, skip this step. What I mean by this step is that experience really matters. The more testers you have watched, the more basis of comparison you have to know the difference between normal and not-normal work.

2. **Put yourself into a watchful, sympathetic state.** Be open to what you are shown and NOT shown. You should be sympathetic because the people you are reviewing will be nervous. Look for good things and praise them, this will help them hear about the not so good things.

3. **Witness the testing.** Therefore, say "Show me your testing."

   If the tester immediately begins to test the product, right in front of you, good. You can begin to evaluate that. But usually, you will not be shown testing. *Usually* the tester will wave documents at you that describe testing or tools that help perform testing. So, what you have to do next is drill down.

   Say "Show me exactly how this relates to the testing." Or, if you were shown a test procedure or test support document, you can point to any piece of it and say "what do you actually do when you read that part? Show me." My favorite tactic as a reviewer is to visualize the testing, just as if it were a movie in my head. I want to see where the tester is, what he's looking at, what's on the screen, and the precise way he interacts with any tools or artifacts. I listen for hesitations and vague speech, and wild claims, too.

4. **Whatever you are shown, relate that to all the major parts of your mental model of testing.** The most basic model of testing I use is called the Heuristic Test Strategy Model and has these main elements: project environment, product elements, quality criteria, test techniques. Embedded in those elements are such vital concepts as oracles (how you recognize bugs), coverage (what you test), and procedures (specific things you do to test). Surrounding and permeating all of that is risk.

5. **Engage the tester in conversation to evaluate how he relates the work to his own model of testing**. It's usually not enough to test well. You also have to explain how you are testing well. So, I make that a big part of the evaluation. This is also part of building a rapport and a working relationship with the tester.

In a spot review I am not going to do everything. I drill down at a variety of points and look for trouble. Trouble means anything potentially harmful to the business, such as the wrong things are being tested, or the actual testing doesn't match the described process, or the testing is missing potentially important bugs.

It was great fun. I got challenged, too. Pradeep, head of the lab, asked me to pair with him to analyze a product and create a test strategy while a dozen of his testers looked on. At the end of it, one of the onlookers showed us a mind map he made that described our process. Then DS dared me to face a job interview as if I wanted to join his team. Parimala arranged to pick me up and drop me off from the hotel each day so that she could badger me with questions.

The Tiger Cub Problem

Moolya is a test lab full of the life of mind. Long may it prosper. But that will come down to one person, I think: Parimala. Pradeep put her in charge of training. This is a key role. The biggest challenge growing a test lab, apart from dealing with unreasonable clients, is assuring that the testers actually know how to test. In Moolya's case they want to do something unprecedented—they want to be ready for any client to challenge their expertise. Most test labs advertise that they have expertise. Try asking any of them to prove it. You will get nowhere. Moolya claims to be ready for that question, and so they need to BE ready.

This means Parimala must create a training and mentoring program that prevents any tester from being recognized as excellent until he's put in the time and earned the *grudging* respect of critical-eyed peers. This amounts to a certification program, of course: one that is community-based and also based on demonstrated skill over time. Since the community is growing all the time and there are many sub-skills of testing, that's a lot of work. I think it will take her a good 18 months to get really organized. Unfortunately, she needs to be ready right now. It's a tough gig.

Part of Moolya's strategy is to establish an intellectual culture. To help with this they created a role called a "shifu." The idea comes from Chinese martial arts. A shifu is a master. A shifu at Moolya represents a tester who is considered fully capable of representing the lab as a tester and also trains other testers. It is vital to make it difficult to become recognized as a shifu. It must be an honor, and the testers in the lab must believe that the shifus actually are the lab's best experts, or else they will not be motivated to join them.

I think this strategy is crucial, but it's also quite difficult to pull off. Apart from the problem of determining a fair way to identify these worthy testers, a test lab must contend with the *tiger cub* problem. I have also explained this to Mindtree and Cognizant, in briefings with them. The Tiger cub problem is the main reason that no test lab, to my knowledge, has created a systematic program of expertise building since I did it when I worked at STLabs in the mid-90's. You see, if you get a tiger cub as a pet, it's cute at first, but it grows up to be dangerous. Testing experts are like that, too. At STLabs, we once had a half-day strike of all the test leads to protest certain management misbehavior. I was so proud of them.

When I worked at Reliable Software Technologies, years ago, I would refuse to work on any client project that I judged to be fundamentally broken. My company wanted me to bill hours, but I felt I could take money from a client if I felt he was acting from impaired judgment and against his own best interest. I quit that job after six months—just before they fired me.

The Rise of the Intellectual Indian Software Tester

Part 3

- by James Bach

[Please read part 1 and 2 of this series in previous issues of Tea-time with Testers]

India. Yeah! But first, I need to talk about training testers, because that's a special problem in India.

Training is sometimes assumed to be a process of transferring knowledge—as one might deliver a package from a truck. Actually, that model does work for simple things. If I "teach" you my phone number, I'm essentially dropping it at your front doorstep. But for complicated, subtle, or otherwise difficult to practice skills, it can cause a lot of harm to think of teaching that way. A student who has any trouble learning, and yet believes learning is just a matter of "taking in" a delivery of "knowledge" will soon begin to feel stupid, and stop even trying to learn.

The trouble is, as a teacher, I can't actually open your mind and pour the knowledge in. Your mind can't open that much and the knowledge is too big. So, at the very least, teaching is like building a ship in a bottle, piece by piece.

No, it's not like that either, because a bottle is just a space surrounded by a barrier. Your mind is a different kind of thing. There's no "empty space" in it. Every space is already taken. To learn is to create a new space and connect it to all the other stuff in all the other spaces. It's a construction process, and no teacher can do that, only the student can. Therefore, all teaching is a process of helping the student's own learning process. All learning is self-learning with various degrees of assistance and stimulation from the outside. It's like doing surgery on yourself, while the surgeon tells you what to do.

And, no, I still haven't got it, because no one can tell you how exactly to construct yourself. And there's more. Learning is also a process of adjusting, or unlearning, lots of outdated or broken old ideas. Learning is also a process of changing your identity; slowly coming to feel like an expert. It's also a process confronting, challenging, and sometimes accepting your own limitations. Learning is personal and social; tacit and explicit. It is mental and physical and emotional. Whew!

So, a teacher helps… somehow… in an indirect way. But learning is mostly about the learner and under the learner's control, whether he realizes that or not.

My Teaching Method

I find it useful to a baking analogy to describe my particular indirect method of teaching. A baker applies energy to create chemical changes, otherwise the bread doesn't happen. The energy I work with is personal enthusiasm, nurtured through experiences of enjoyable success in testing.

1. **Attract people who might want to learn testing, and filter out the people who probably won't learn from me. ("Start with the right ingredients")**

a. Be excited about what I do and public about how I do it, so that right students find their way to me, and the right ones stay away.
b. Make them think I care about them by using the tactic of actually caring about them.
c. Have high *aspirations* for each student, but keep *expectations* low at first.
d. Always tell the truth about how they are doing, but work a little harder to find hopeful news than to find discouraging news.

2. **Acknowledge and begin to disarm their secret fears ("get rid of any roaches in the kitchen"):**

a. That they aren't smart enough to test.
b. That testing is a waste of what talents they do have.
c. That testers are unlovable.

3. **Get them hooked on testing excellence ("start the fire") using the following tactics:**

a. Focus first on the thrill of discovery.
b. Give them an experience of the joy and power of their own curiosity.
c. Demonstrate the connection of testing to something they already know, so that they don't feel like hopeless beginners.
d. Provoke them to test intuitively, watch for moments of talent, then reveal evidence of their talent to make it tangible for them.
e. Give them a sense of what mysteries are revealed to those who study testing deeply.

4. Challenge their naïve beliefs about testing ("clean the kitchen") using the following tactics:

a. Bring their mental models of testing to the surface through conversation and practical exercises.
b. Honor the good intent and any grains of truth to be found there.
c. Use practical exercises, Socratic coaching, or direct explanation (in order of preference) to reveal weaknesses in typical folk beliefs about testing.
d. Demonstrate and explain powerful alternatives.

5. Give them permission to reinvent testing for themselves, ("let them cook") and invite them into the community of ethical, creative software testers.

a. Get the student to try to solve a problem before telling him how to solve it.
b. Teach the identification, development, and proper use of heuristic methods.
c. Expressly invite the student to create modified versions of models and tools.
d. Demonstrate the process of context-driven process engineering.
e. Remind them "if you only know one way to do something, you are a robot, not a thinker."
f. Discuss ethical challenges and share stories about way to meet those challenges.
g. Celebrate heroes and acts of heroism. Offer credit where credit is earned.
h. Interact on public forums. Regularly refer to people we respect.
i. As a teacher, discuss my own struggles and ethical regrets.

6. Challenge them to practice the difficult aspects of testing, under various kinds of pressure ("bake the bread"):

a. Hold peer conferences to examine and compare experiences.
b. Provide and receive coaching.
c. Practice with documentation.
d. Practice with tools.
e. Practice with mathematics.
f. Practice with written and oral reports.
g. Practice under time pressure.
h. Practice under critical attention of peers.
i. Practice in public.
j. Practice with unfair project conditions.
k. Practice dealing with failure.
l. Practice in unfamiliar contexts.
m. Write about these experiences.

7. Acknowledge and celebrate successes, improvements, and boldly achieved failure ("enjoy the feast and thank the cook").

What About India?

I was getting to that… My point is that having good idea, even the "best" idea, is a small part of getting other people to have good ideas, too. Just knowing how to test does not in any way make us good teachers of testing. No wonder that in all the years software testing has been a commonplace and accepted part of software development, it has barely advanced as a recognized craft. No wonder the ISTQB can package outdated and demonstrably wrong testing folklore into a syllabus and successfully make people pay to get "certified" in it. Testing is hard to teach, it's hard to observe, and that makes us easy targets for fakers and quacks who want to make a buck.

Most of the testing world remains, to this day, in a dark age. But there is hope. The testing industry cannot be changed from the outside. And no one person can force it to be better, even from the inside. But small groups of testers who understand community dynamics and *who work on themselves first*, can drive a process that eventually reinvents the testing craft. This is already happening.

I believe if the Indian testing industry is to become vastly more productive than it is now—and more respected—its practitioners and clients must profoundly change how they think and talk about testing. They must leave the factory and manual labor metaphors behind. They must embrace testing as a process of skilled, creative design. They must come to value effective critical thinking, not key presses per second or test cases executed per hour. They must refocus their process improvement efforts on three things: developing skills (rather than procedures), developing heuristics (rather than standards), and developing supportive tools (rather than tools that alienate us from our own testing process).

That's why I described my teaching method, above: my theory is that the same basic approach works whether you are developing yourself, or helping *whole countries of testers* to develop.

Notice step one of my method. It's not about explaining anything. It's social. It's about attracting the right people and repelling others. Yes, the craft can be improved through a direct assault on bad ideas and the people who hold them. You can stand on a chair and shout, but most people won't listen to you in the right way unless they feel you are part of their community. Social and personal factors dominate rational factors in humans every time.

So the rise of the intellectual Indian tester is going to happen mostly along the line of social connections. It begins with a small community, but they will attract more, by virtue of their reputation for excellence and the inherent pleasure of testing with our minds.

Notice step two. We have to deal with the fears of the testers who may want to adopt this new way of thinking about testing. Partly, there is a perception of safety in numbers— and that's what will keep most testers in the other camp of factory-style testing—but over time this will shift. Meanwhile, we need to offer what support we can on the forums, in conferences, and peer to peer. Just tonight I was Skyping with yet another Indian tester who needed a pep talk for how to deal with a bad manager. We must make it part of our job to help each other that way.

For step three we need public outreach in the form of testing parties and open forums. That's the fun part of growing community.

Step four can be a bit messy. Challenging bad beliefs is best done within the community of committed intellectual testers, but it can be difficult to arrange that in any sort of public forum. An excellent way to do it is to hold peer conferences—which are small gatherings of invited testers who come together to share experience reports and then to face hard questioning from their peers. Again that's going to work on a small scale, and for people who are already committed enough to come to a peer conference. What about the rest?

Some of my colleagues suggest avoiding public debates. I'm famous for starting arguments on Twitter with people I feel are bad testers. It think it's important to do some of that, at least, even though debating is often not much fun. It's important because otherwise new testers will think that the Factory Schoolers own the craft. I understand why a lot of my colleagues want to be nice to testers and consultants who push ISTQB certification, "100% automation" and other silly ideas. They think being nice will change the prevailing practices in the world. On the other hand, I've seen this play out over the last 25 years and it seems to me that being nice hasn't worked *even a little tiny bit*. Working with the ISTQB people did not stop them, it helped them.

I will say this: if someone is sincerely trying to learn testing, be friendly to him. Otherwise, don't waste your time hoping that charm will win the day. In that case debate is the better path. Let's shake things up. Let's get louder.

The rest of the steps also relate to the transformation of Indian testing, but I'll leave them as an exercise for the reader, at least for now. It's time to talk about the last part of my India trip.

My Visit to Barclays

The same dynamics work for developing a company culture of skilled testing, too. Companies must hire carefully and have some method for ridding itself of people who aren't getting the job done; they must drive out fear; they must engage workers minds, and not just assume that paying them creates all the motivation required. All these things map to the steps I outlined.

I've been working with Barclays Bank for a couple of years now. They are going through those steps, more or less, as a corporate testing culture. It's a big organization, and big organizations can't change very quickly. They have a substantial number of testers working in India, now, which is why I came to Pune.

When I first worked with Barclays in Singapore, they selected a relatively small group of very ambitious testers for me to work with ("the right ingredients"). I began their training, and then they took the initiative to teach a version of my material to many others after I left. They created that special version of Rapid Software Testing training by themselves, modified from my materials ("let them cook"). They established several internal forums for discussing and encouraging the learning ("starting the fire and cleaning the kitchen"). Over time, some of those testers have come a long way ("baking the bread"), and one systematic approach they used is a testing competition spread over a year.

Changing an existing organization of 700 testers while they are working on projects everyday is a slow process. They've been working at it a couple of years now, though, and it's coming along well. But there's a new twist: Barclays now has a large contingent of Indian testers, and Keith Klain, the leader of their Global Test Center, wants to create the same skilled testing culture there as in Singapore and his other sites.

I thought that might be a little challenging. It really wasn't. Teaching my class at their Pune site, I was once again impressed with the high quality of the students. They were young, ambitious people who dived into the material with enthusiasm. The right ingredients.

Afterward I judged the last part of the 2012 "Super Tester" competition. The task for the finalists was to test XMind using tools. We didn't specify how the tools should be used, only that the testing would be judged on the basis of effectiveness and originality. This is an example of "baking the bread"—testing under time pressure and scrutiny. Here the result was a little disappointing: because none of the testers did much with tools. I had suspected that the testers may not be comfortable with tools, and that's why I wanted to give them this challenge. Now they know they need to practice harder to become more comfortable with the use of tools in testing. This is why it's important to push each other. We must discover our weaknesses so that we can deliberately work on them. A testing competition is a relatively safe place to do that.

The Challenge For India

What is an intellectual software testing culture? It's people who feel comfortable and eager to face any complex technical and philosophical problem that comes up in their projects. It's people who understand that excellent testing is difficult to do, and who take responsibility to educate not only themselves, but also their clients and colleagues about how it can be done. It's a self-perpetuating social order of skilled testers who are valued and supported by the intellectual software developers with whom they work.

I visited three other large companies in India, before I left. I saw a lot of what I expected, and some things I didn't. I didn't expect much enthusiasm for exploratory testing, for instance, but instead there was quite a lot. Everybody I talked to wanted to know more about it. The spark of an intellectual testing culture is there.

What surprised me a little are the questions I was asked about exploratory testing: How do you measure it? How do you document it? How do you estimate it? I was asked these questions as if "normal" testing is easy to measure, document, and estimate. Of course it isn't. Of course the fact that you think about and develop testing as you go rather than write it out beforehand does not in any way make it harder to measure, document, or estimate. Testing is always hard to measure, etc. Scripted testing *seems* easier when it comes to those things, because scripts are visible. Except that scripts aren't testing.

These questions come mostly from management. That's the challenge, then. Management in India has to get serious about testing. They need to learn how to test, and break out of their illusions about counting little test cases as if they were units of testing goodness when as I just said, scripts aren't testing. As I wrote in part two, Moolya has figured this out. Parts of Barclays have figured this out. But management in the rest of India seems to have a long way to go.

Back To Index

James Marcus Bach is a software tester, author, trainer and consultant. He is a proponent of Exploratory testing and the Context-Driven School of software testing, and is credited with developing Session-based testing.

His book "**Lessons Learned in Software Testing**" has been cited over 130 times according to Google Scholar, and several of his articles have been cited dozens of times including his work on heuristics for testing and on the Capability Maturity Model. He wrote numerous articles for IEEE Computer.

Since 1999, he works as independent consultant out of Eastsound, Washington.

He is an advisor to the Lifeboat Foundation as a computing expert.

Follow James on Twitter @jamesmarcusbach or know more about his work on satisfice.com