

THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

JANUARY 2013 | YEAR 2 ISSUE XII

Jerry Weinberg

Intelligence or Problem Solving Ability

Johanna Rothman

What makes a Great Tester?

James Bach

The Rise of The Intellectual Indian Tester –part 2

T Ashok

Too Many Conditions

Joel Montvelisky

Of Testers and Soldiers...

Issachar Hazan

Dealing with Stress

There are more than 40 leading
organisations that host
Tea-time with Testers
on their knowledge portal.

WHAT ABOUT YOURS?

TEA-TIME WITH TESTERS

Subscribe [here](#) Right Away to get our all Issues for FREE



TEA-TIME WITH TESTERS

First Indian Testing Magazine to reach 101 Countries in the world !

**Created and Published
by:**

Tea-time with Testers.
Hiranandani, Powai,
Mumbai -400076
Maharashtra, India.

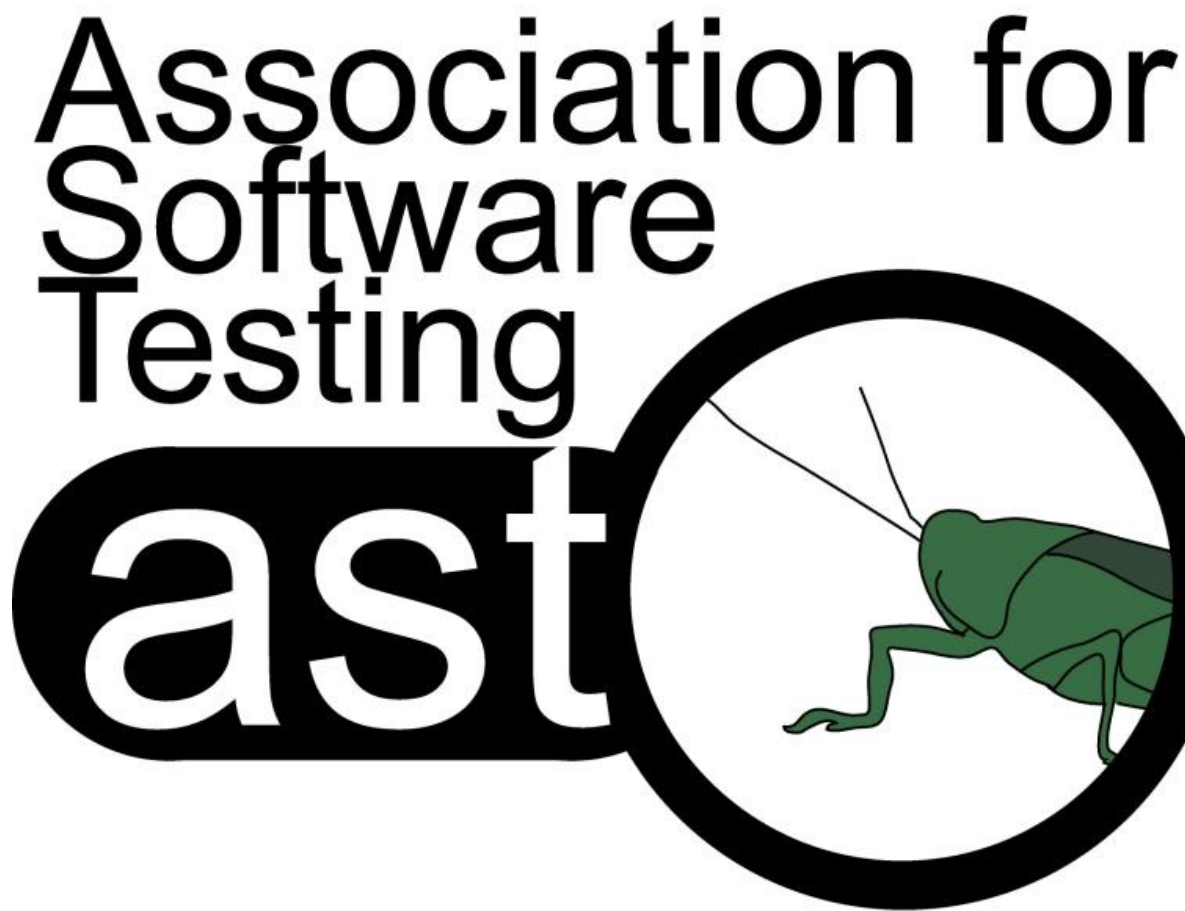
**Editorial and Advertising
Enquiries:**

Email: editor@teatimewithtesters.com
Pratik: (+91) 9819013139
Lalit: (+91) 8275562299

This ezine is edited, designed and published by
Tea-time with Testers.

No part of this magazine may be reproduced,
transmitted, distributed or copied without prior written
permission of original authors of respective articles.

Opinions expressed in this ezine do not necessarily
reflect those of the editors of ***Tea-time with Testers.***



is sponsoring its first “AST Test Forum” in **PUNE, INDIA** on **February 13th, from 5 – 7 pm** at the **IDB AUDITORIUM, COGNIZANT PHASE I, HINJEWADI.**

“The forum will include an introduction to the AST by current board member **Keith Klain**, an open Q&A session, and a chance to network with testing professionals from different industries.”

REGISTER FOR THE FORUM [HERE](#)



Editorial

Wasting or Investing ?

It all started with my discussion with a friend about *debrief sessions* that we follow as part of Session Based Test Management. When I asked if he knew about it already he said that he never heard that word before. Then I explained him what we typically do in those debrief sessions and asked him for his opinion.

Looking at his expressions I was little doubtful if he really understood what I meant but without losing a minute he replied, "That's absolute wastage of time. I would rather ask my team to utilize those 20/30 minutes in their testing rather than wasting them in this *debrief* thing".

Initially I got surprised. Someone called *that* thing as *wastage* of time which I'll always call as best investment. Well, that was his thinking and his opinion. What made me write this post are these two thoughts:

1. 'Can there be other things in testing that I (or anyone else) find as good investment but someone else finds them as complete wastage of time?' What would one call 'test design using mind-maps', or say 'creating charters'? Wastage or investment?
2. Or is it like people just form negative opinions about things that are *new* to them & that will *require* them to study the craft?

Well, I am still trying to find answers. What do **you** think by the way? I will wait for your letters.

Until then...

Yours Sincerely,



- **Lalitkumar Bhamare**

editor@teatimewithtesters.com



QuickLook



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

The Rise of the Intellectual Indian Tester - 19

In the School of Testing

What Makes a Great Tester? -30

Dealing with Stress - 35

Of Testers & Soldiers... - 38

T' Talks

Too many conditions! - 43

Testing Puzzle – S.T.O.M. Contest

Our Testimonials

Family de Tea-time with Testers



What's making News?

- find out the latest happenings in the technology world

Student checks software for critical bug, gets expelled from college

When 20-year-old Ahmed Al-Khabaz, a computer science student at Montreal's Dawson College, discovered a critical flaw in his college's student web portal, he decided it was his "moral duty" to share the discovery with the institution's leaders so that the bug can be fixed before doing serious harm.

But what he probably could not have imagined at the time is that this – for all intents and purposes – honorable decision will ultimately lead to his expulsion from college.

Al-Khabaz, who was also a member of the college's software development club, and fellow student Ovidiu Mija were working on a mobile app that would facilitate the students' access to their account on the portal in question, when they discovered that the web application's "sloppy coding" allows anyone with a basic knowledge of computers to access all of the student's accounts and the information contained in it: personal information (including Social Security numbers), grades, class schedule, and more.

They shared what they discovered with François Paradis, the college's Director of Information Services and Technology, and he seemed satisfied with the discovery. He promised to talk to Skytech, the firm that created the Omnivox portal and online services platform, and have them fix the flaw.

It could all have ended here, and Al-Khabaz would still be a student of the college, had he not decided to check whether the flaw was fixed and whether he could find other crucial vulnerabilities by pointing the Acunetix Web Vulnerability Scanner – a legitimate piece of penetration testing software that automates some of the most popular attack techniques against web applications – towards the Omnivox web portal.

A few minutes after initiating the “attack”, he received a phone call from Skytech President Edouard Taza, who told him to stop what he was doing.

“I apologized, repeatedly, and explained that I was one of the people who discovered the vulnerability earlier that week and was just testing to make sure it was fixed. He told me that I could go to jail for six to twelve months for what I had just done and if I didn’t agree to meet with him and sign a non-disclosure agreement he was going to call the RCMP and have me arrested. So I signed the agreement,” Al-Khabaz shared with National Post.

Courtesy: [Net-Security](#)

Click [here](#) for complete news.

Are you interested in publishing the news
about your own firm, tools, community and
conferences in **Tea-time with Testers?**

Then write to us at:

contact@teatimewithtesters.com

with “**News Enquiry**”* in your subject line.

*Conditions Apply



Want to connect with right audience?



How would you like to reach over **19,000** test professionals across **101 countries** in the world that read and religiously follow "Tea-time with Testers"?


How about reaching industry thought leaders, intelligent managers and decision makers of organizations?

At "Tea-time with Testers", we're all about making the circle bigger, so get in touch with us to see how you can get in touch with those who matter to you!

ADVERTISE WITH US

To know about our unique offerings and detailed media kit

write to us at sales@teatimewithtesters.com



Announcing...

Smart Tester of the Month Award !!!

❖ Winner for Puzzle:

Sharaniya Srinivasan

Congratulations!

Announcing TTwT 2013 Free Gift Calendar

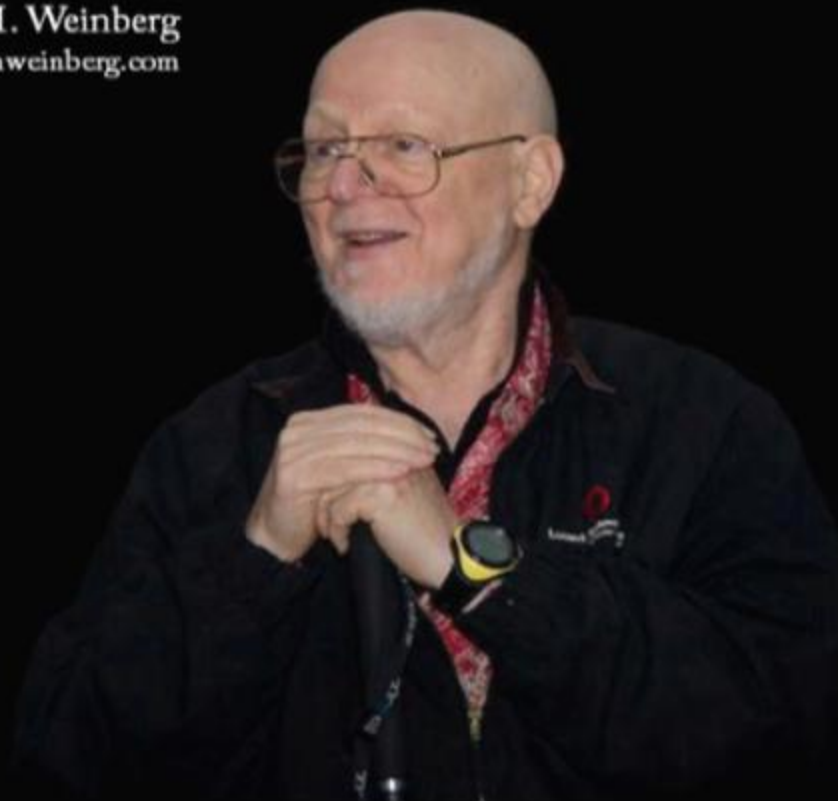


"Your ideal form of influence is first to help people see their world more clearly, and then to let them decide what to do next."

- Gerald M. Weinberg
www.geraldweinberg.com

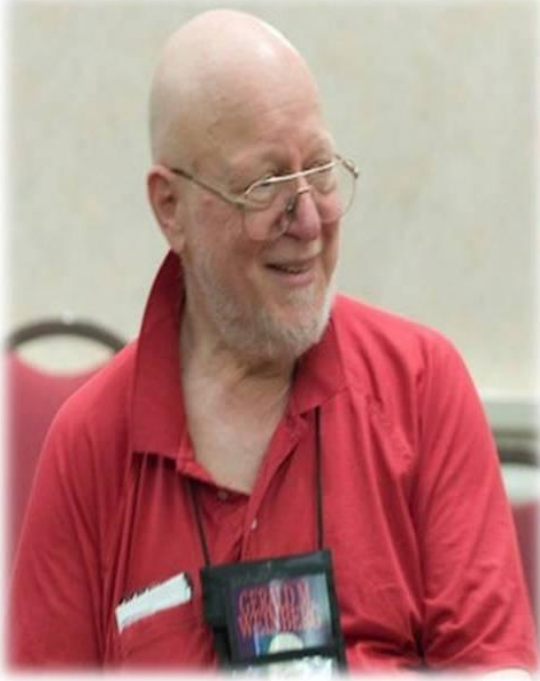
January

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		



Click [HERE](#) to download

Tea & Testing



with

Jerry Weinberg

Intelligence, or Problem-Solving Ability (Part 3)

APTITUDE TESTS

If intelligence is so important for programming success, what can be done to select those people who have what it takes to do the job? The possibility of administering tests to select programmers had long bewitched programming managers, but over the years nobody has ever been able to demonstrate that any of the various "programmer's aptitude" tests was worth the money it cost for printing. We should be remiss, however, if we did not attempt to explore the reasons for this universal failure.

In the first place, of course, programming is a diverse activity, and any test that gives a single "grade" is hardly adequate to measure the aptitudes required. Some tests have been designed to give multiple scores, but even if these had individually valid measures, they would be difficult to apply sensibly, and the average personnel manager simply would not bother.

Moreover, just because of the multidimensionality, someone who is sorely deficient in one area may turn out to be outstanding in another—provided that the opportunity to use his strong points exists. On the other hand, sometimes a person who scores high on every possible scale performs poorly because he is missing some "minor" ability, such as the ability to get along with the people he has to work with. These are theoretical conjectures, however, because nobody has put together a test with any measurable validity. The closest thing we have to a validation of a programmer's aptitude test are

studies which show that people who scored high on the test were better students in the ensuing programmers' training. When it got down to the nitty-gritty of actual programming, however, there was no correlation—or even a slight negative correlation—between test scores and rated performance.

This sorry picture is not unique to programming. Intelligence tests generally—such as the famous IQ tests—are able to predict success in school-type situations—and in nothing else. In fact, as one wit put it, intelligence tests measure the ability to take tests. We have reason to believe that this appraisal is not far from the truth. For example, in IQ tests, speed is very much a factor, as it is in school situations generally. But, in the office, the difference between one hour and one hour and ten minutes is only ten minutes—not the difference between an A and a C. Slow and steady may not win the race, but programming is not a race.

Another typical distortion of intelligence tests is in the emphasis they place on short-term—rather than long-term—memory. They could hardly be expected to do otherwise, given the constraints under which they must be administered. In an IQ test, one is asked to memorize nonsense words or arbitrary lists. But in "real life," it is selective memory which pays—the ability to forget the unimportant and retain the important over long periods. Not so in school, however, and so we have the IQ test and the school grades going hand in hand—off in a different direction from anything we are interested in.

Finally, and this leads right into our next topic, IQ scores, and programmer's aptitude scores, are demonstrably correlated with certain forms of training. In big cities, an eager parent can take his child to a special tutor who guarantees to raise his IQ by so many points for so many dollars. The same techniques are used by certain programming schools to help their graduates get jobs with those companies that rely heavily on aptitude testing. So, no matter how much we would like to have a magic wand which would point out the best programmer prospects, we are just going to have to learn to do without.

APTITUDE TESTS FOR PROGRAMMING

All of this leads up to specific tests that have been used in a grand attempt to measure aptitude for programming.

Probably the foremost among these is the so-called PAT, or Programmer's Aptitude Test. Actually, this is not a single test, but an unknown number of variants on an original test made up one idle afternoon by a group of programmers in IBM's New York Scientific Computing Center sometime before 1956, and administered there to all job applicants and interested visitors. One of the reasons for the profusion of variants is that the test has been so widely used and so unprotected that nobody knows how many people have taken it, or, in particular, which people have taken it.

Not that the variations give much protection, since there is considerable transfer of learning from one variant to another. Over a series of classes involving IBM and other programmers, I asked people to report the number of times they had taken a variant of the PAT and what their scores had been. Since this was a "safe" situation—with nobody's job at stake—there is reason to believe that the replies are not too inaccurate. Out of 27 people who had taken the PAT more than once, 23 received A's the last time they took the test—they were all employed programmers working for companies which regarded A's as an important hiring criterion. Of these, 12 had A's the first time they took it, 7 had B's, and 4 had C's. Nobody had done worse the last time than the first.

A typical case was a woman working for IBM as a programmer who had graduated from college with a Math major and applied to RCA for a programming job. They had given her the PAT, and she had scored a "low" B. She didn't get the job, so she interviewed with IBM, who administered a slightly different

version of the test, on which she scored a clear A. She was asked whether she had ever taken the PAT before, and she (intelligently) said "no," whereupon she was hired.

This previous experience with the PAT may be one reason researchers have been unable to correlate PAT performance with job performance. A few correlations have been reported (see Reinstedt, 1964}, but we must be careful as to what we regard as "significant" in these cases. For example, the best correlation found in these studies was 0.7 between the PAT and supervisor's ranking. What does 0.7 correlation mean?

For a single study, a 0.7 correlation between two variables means that $(0.7)^2 = 0.49$, or 49 percent of the variation in one variable can be accounted for by the other—although it doesn't say which is which. This still leaves more than half of the variation to be explained, even if the correlation coefficient were an "explanation." One of the reasons for the popularity of the correlation coefficient is the way it seems to overstate the case, since the number used is always larger than its square.

The second problem with such correlations is that the score—which is definite enough—is correlated with the supervisor's ranking, which is an uncertain measure of programmer performance, to say the least. There really is no instrument today for measuring programmer performance—or reducing it to a single number, in any case. Consequently, there is nothing really reliable with which to correlate the PAT. It may even be that the PAT is a marvelous predictor of programmer performance, but the supervisors are not themselves sufficiently trained to know it.

We must also be wary of one other thing in using such correlations. Nobody knows how many times people have tried to correlate the PAT with job performance. Those who did try and did not obtain "significant" results more often than not would not publish their trial. We know of similar trials because correlations have been found between test scores and school grades (Biamonte, 1964, Gotterer, 1964). But, you see, if given sufficient chances to correlate, we will eventually get some correlation coefficient above any level we desire, if the data are random. Thus, we could say that a correlation of 0.56 will arise, by chance, no more than one time in a hundred, but if we do not know how many trials have been made and not reported, we have no way of evaluating the significance of such a statement. We do know, however, that the PAT is used in hundreds of places.

Furthermore, even if the trials have only been made a few times, a correlation of 0.56 occurring one time in a hundred by chance assumes that the true correlation is zero. If there is a small positive correlation, say of 0.1, then a spurious measure of 0.56 will occur more frequently than one time in a hundred. But a correlation of 0.1 means that 1 percent of the variation is accounted for by the correlation, and this is hardly the type of information we can use to make personnel decisions.

Assuming that the correlations were reliable, we have, on the basis of the recorded literature, no instrument any better than the PAT. Admittedly, it does predict scores in programming classes fairly well, but that is not what we are buying when we hire a programmer. As Mayer (1968) puts it so well, Very probably if you would use all of the tests to select an individual, you can [sic.] obtain a person who has a high probability of successfully completing your training program. Whether this individual is going to like programming or will possess the motivation that will allow him to take the successful training onto the job site is a question that is not yet answered.

Even this condemnation implies that the tests may be all right as far as they go, but that other factors may be more important on the job than pure "aptitude." Although we can't quarrel with the conclusion, we think a possibility may have been overlooked—namely, that the tests are just not good ones for programmer aptitude. Given the history of the PAT, one wonders why so many hundreds of firms use it slavishly, year after year. A little examination into the structure of the test itself might give us some hints as to what is wrong with it, now that we know that much is wrong with it.

The original PAT had three sections: relationship rules for geometric figures (much like standard IQ series), arithmetic reasoning, and number series. The first two have been retained in almost all of the variants of the PAT, although the number series has often been replaced by letter series. Just to get an idea of what might be wrong, consider the letter or number series problems, a typical one of which might be (1 4 7...) where the examinee is asked to supply the next number in the series. It is certainly plausible that a modicum of intelligence is required to answer "10", but is this really the ability we most need in a programmer?

Let me give an example of what a good programmer did with such a series. A FORTRAN program had been written with the statement `DO 15 I = 10000, 30000, 10000`. The program went into a strange loop which finally ended but produced output that nobody could decipher. When it was brought to this programmer, he thought for a while and then asked himself under what circumstances this particular `DO` would not produce the series 10000 20000 30000 . . .

Suddenly he realized that if the numbers were being kept in a 15-bit register, the next number in the series would be 7232, and the series would look something like this: 10000 20000 30000 7232 17232 27232 . . . which explained precisely what was wrong with the program.

In other words, the PAT tests for the ability to see the conventional pattern in a series of numbers or letters, but the programmer, especially when debugging, has to look precisely for the unconventional pattern—the deviations. Possibly a much better type of question for programmer aptitude would be something like this:

"Given the series (1 4 7...), tell what might normally be expected as the next number, and then describe at least three other choices which might be correct, giving reasons for each choice."

Another section of the PAT is arithmetic reasoning, but I have never had anyone who has been able to explain to me why programmers have to be good at arithmetic. Perhaps before 1956 arithmetic was more important, when most programming was done much closer to machine language than it is today. If you are working in a relatively crude language, it is useful to be able to add two or three hexadecimal numbers so that you can find an address in a dump, but how many programmers do that today? Don't we have computers just so we don't have to do arithmetic? I myself have never been able to add 7 and 5 rapidly, but I don't think that is the thing holding me back as a programmer. In fact, knowing I am not very good at arithmetic, I am more suspicious of my programs, and of my check calculations—which forces me to do more testing of them. It would be rather easy to make an argument that poor arithmetic ability is an asset to a programmer.

The third part of the PAT leaves me entirely befuddled. Even in 1956, geometric relationships never seemed to have much to do with programming aptitude, but perhaps I missed the point. The one thing that programming doesn't seem to be in today's world is geometric. I've never met a programmer who was asked to tell whether two programs were the same if one was rotated 90 degrees. (There was once an article about palindromic programs, which read the same forward and backward, but its contribution to the profession was minor.)

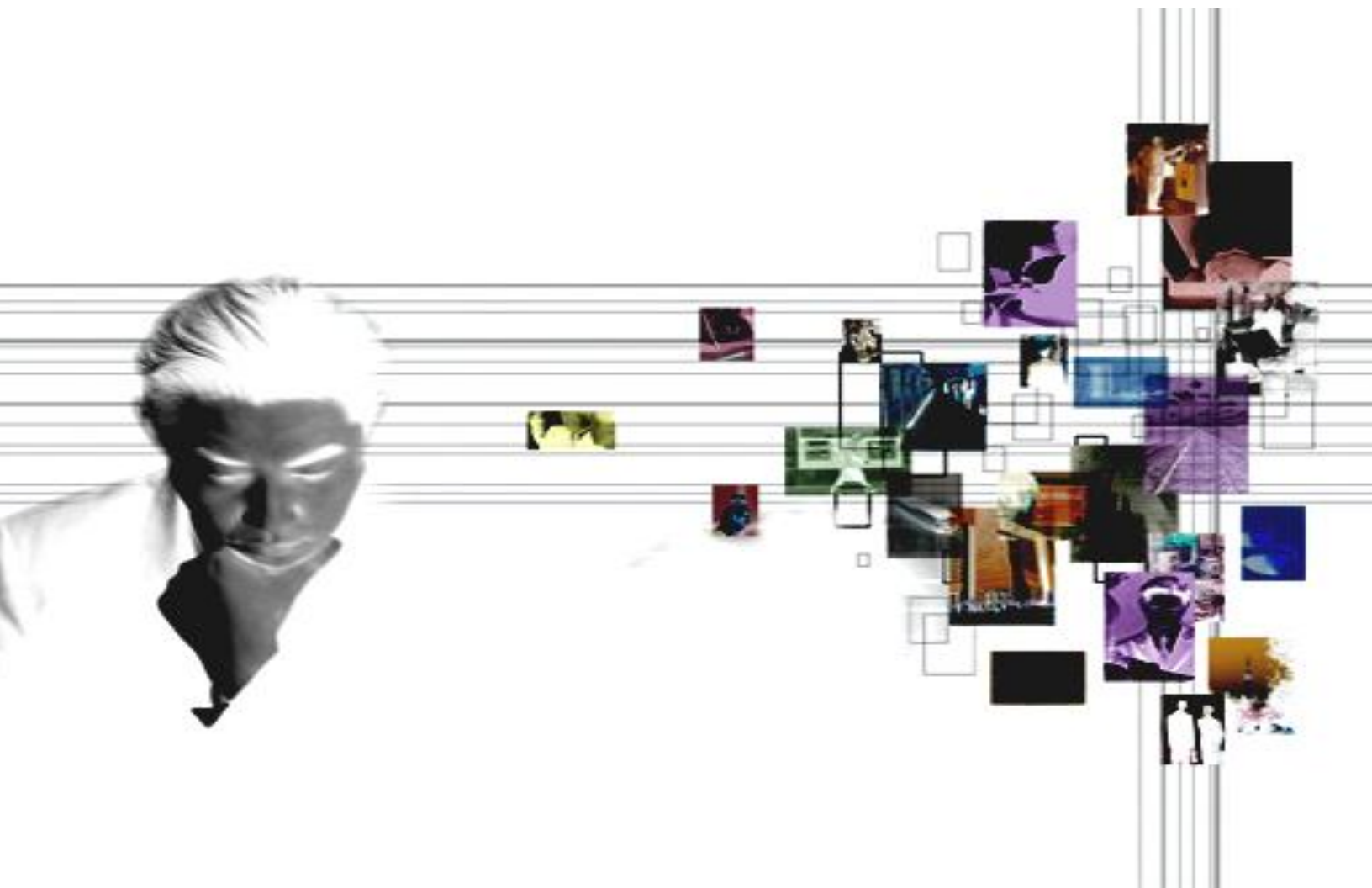
Perhaps it is time that some new thinking go into these aptitude tests, if people are going to persist in using them. And persist they will, since even if the promise of success is not great, the rewards are. So let me suggest a few things that might make more sense for identifying potentially successful programmers:

1. Give the examinee thirty or forty papers with random printing on them and ask him to place them in ten boxes. One week later, ask him to find the paper with the word "COMMODITY" on it. Time his retrieval performance—giving high scores for fast work.



I don't really have much hope for such tests—either that they will work or that they will be applied if they do work—but they certainly seem more promising than what we have been using so far.

Yet all is not quite so bleak. When we are selecting among experienced programmers, the situation is potentially different, although few authors or employees seem to realize it. For example, out of 282 organizations using the PAT in one survey, 138 of them still use it for selecting experienced programmers. Why? Because they feel they have nothing else. Lacking anything better, they try what is available in a vain search for the elusive magic test. Such companies are sitting ducks for anyone who comes along with a fancy package of promises—and with lots of sitting ducks, can the hunters be far behind?

[Back To Index](#)

Biography

Gerald Marvin (Jerry) Weinberg is an American computer scientist, author and teacher of the psychology and anthropology of computer software development.



For more than 50 years, he has worked on transforming software organizations. He is author or co-author of many articles and books, including *The Psychology of Computer Programming*. His books cover all phases of the software life-cycle. They include *Exploring Requirements*, *Rethinking Systems Analysis and Design*, *The Handbook of Walkthroughs*, *Design*.

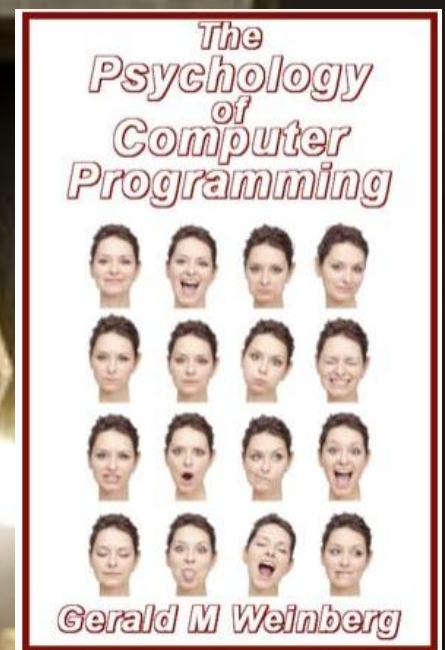
In 1993 he was the Winner of the **J.-D. Warnier Prize for Excellence** in Information Sciences, the 2000 Winner of **The Stevens Award** for Contributions to Software Engineering, and the 2010 **Software Test Professionals first annual Luminary Award**.

To know more about Gerald and his work, please visit his Official Website [here](#).

Gerald can be reached at hardpretzel@earthlink.net or on twitter @JerryWeinberg

Jerry's another book **The Psychology of Computer Programming** is known as the first major book to address programming as an individual and team effort.

"Whether you're part of the generation of the 1960's and 1970's, or part of the current generation . . . you owe it to yourself to pick up a copy of this wonderful book." says **Ed Yourdon, Cutter IT E-Mail Advisor**



TTWT Rating: ★★★★★

Sample of this book can be read online [here](#).

To know more about Jerry's writing on software please click [here](#).

A photograph of a green, conical pendulum bob hanging from a thin wire. The bob is positioned over a surface of light-colored sand. In the sand, directly beneath the bob, is a faint, circular mandala-like pattern drawn with fingers. The entire scene is framed by a dark blue border.

Speaking Tester's Mind

- straight from the author's desk



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 Moolya¹.

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 Pradeep² 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 pabulum 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 more reputation a tester has, the more pride he has in himself, the more he needs to protect that reputation. Over time, it becomes very difficult to push these testing tigers into bad projects. If you want an easy management situation, stick with mild-mannered house cats. I'm a grown tiger. Now I run my own company. I only have to answer to my wife, and she really likes me.

For any test lab larger than a couple of people, turning down work from otherwise willing clients is very hard to do. But a great many companies that seek out consulting services are dead wrong about what they need, and part of the ethics of expertise is—just as with a doctor—not to prescribe a medicine to a patient that you know will harm the patient.

Time will tell if Moolya's tiger training strategy works. But I honestly don't see how they can grow much larger, and remain a center for intellectual testers, unless their testers earn their claws.

I have too much to say, so I will need to say the rest in part 3. Having seen an example of a great new spirit in Indian testing in the form of a brand new test lab, I will turn to the challenges facing established companies as they struggle to explore and implement a skilled testing culture.

to be continued in next issue...

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



[Back To Index](#)



Looking for RIGHT job in Software Testing?

visit **Qualityjobsportal.com**

after all, it's your career we are talking about !



Do **YOU** have **IT** in you what it takes to be **GOOD** Testing Coach?

We are looking for skilled **ONLINE TRAINERS** for Manual Testing, Database Testing and Automation Tools like Selenium, QTP, Loadrunner, Quality Center, JMeter and SoapUI.

TEA-TIME WITH TESTERS in association with **QUALITY LEARNING** is offering you this unique opportunity.

If you think that **YOU** are the **PLAYER** then send your profiles to trainers@qualitylearning.in.

Click [here](#) to know more

There was a time when people did not have compass to find right direction. The only guide they had was that guiding star up in the sky.

Do you think that you are also stuck somewhere with technical issues? Do you need help in decision making or want guidance?

Well, the wait is now over . Introducing...

“The Guiding Star”

*The panel of our experts is now here to help you.
Send us your questions around software testing and our Guiding Stars will help you out.*



E-mail your question on –

theguidingstar@teatimewithtesters.com

Please Note :

1. This is not a job portal.
2. Typical interview questions will not be answered.
3. Questions should be on Software Testing or related topics only.

Do you have any Questions or Feedback on articles that we publish in Tea-time with Testers?

No Problemo! We will publish your Feedback/Comments and also the answers to your Questions that you have for our Authors.

Do write us your Feedback and Questions in below format and send it to teatimewithtesters@gmail.com :

- Your Name
- Your Brief Introduction
- Article Name
- Your Feedback or Questions if any

➤ Your Feedback or Question

Make sure to write **Feedback For < Article Name>** in your subject line.



In the school of Testing

for your better learning & sharing experience

A hand is shown holding a red stick figure, which is positioned above two groups of blue stick figures. The blue stick figures are arranged in two pairs, each pair holding hands. The red stick figure is being held by the hand, as if it is about to be placed or is being lifted. The background is white.

Hiring **Great** Testers

[A How-to Guide]

by Johanna Rothman

What Makes a Great Tester?

I bet you would like to hire the “ideal” tester. Well, let me tell you right now, the ideal tester does not exist. That’s because we are all real people, so we are not perfect. That’s the good news. If we were perfect, the world would be a boring place.

On the other hand, we can think about general principles of what makes a great tester, and you can think about what will make a tester great in your culture. Now, you have pretty good criteria for what makes a great tester.

What’s Your Culture?

Before you think about what makes a tester great, think about your culture. You can think about these three aspects of your culture:

- What can people discuss?
- How do people treat each other?
- What do you reward?

These three questions seem pretty simple, don't they? But let me give you some examples before you dash off thinking you know about your culture.

What Can People Discuss?

Many organizations claim they have an "open" culture. "My door is open," many managers say. Well, their doors may be open. But, can you discuss salaries? Can you discuss the criteria for moving to the next level in the organization? Are the testers paid as much as the developers? Do you know? Is there a salary chart?

Money, salary, and expertise criteria—the job ladder—are a primary example of what might be open or not. Your organization might share the job ladders and not share the money. They might share sales orders. They might share product roadmaps. But, they might not.

There is no right or wrong, here. There is only what is right for your organization, your culture.

How Do People Treat Each Other?

The next question is how people treat each other. In some organizations, senior managers are allowed to create emergencies for other people to solve. Or, managers feel as if it's okay to yell and blame people in meetings. Or, vigorous discussion, including *ad hominem* attacks, is how people of all levels disagree with each other. Or, people never come to a decision without everyone agreeing.

I once consulted to an organization where everyone had to agree with every decision. It took that organization a very long time to make any decision at all, including release decisions. Even though they had release criteria, it still took them weeks to decide to release. Everyone had to be happy with the decision. That organization no longer exists.

What do you reward?

Some organizations reward long hours. Some reward hard work. Some reward cleverness. It's hard to tell what others reward because individual managers get to decide.

If your organization is still working in a more traditional project lifecycle, the organization might reward heroes or firefighters. And, the organization might not be very happy with testers who discover problems.

On the other hand, if you have transitioned or are transitioning to a more agile organization, you might discover that the organization rewards people who discover problems earlier.

There is no right or wrong culture. There is *your* culture. Now that you've thought about your culture, let's think about the general principles of what makes a tester great.

What Makes a Tester Great?

When I think about great testers, I think about non-technical skills first—and you might be surprised by that. But that's because it's easy to teach the tools and technology to people with the right background. But as you can see from the culture discussion, it's really hard to find a good match for the culture.

You can't lump all non-technical skills together. I separate them into qualities, preferences, and specific non-technical skills.

Tester Qualities

In my experience, great testers share these qualities.

Testers are skeptical. Testers who don't automatically believe the information from developers and project managers tend to find more problems than the testers who don't challenge the assumptions. Testers who believe developers when they say, "You don't need to test here; there are no bugs here" are the testers who are blindsided when the product is released and the customers find a gazillion problems. The testers who search where the developers say there are no problems tend to find interesting problems.

Testers enjoy discovering problems. Testers find problems in the product; they don't fix product problems. Great testers enjoy discovering problems, whether they automate or explore. Great testers discover ways to use the product that no one intended the product to be used. I am one of those testers. "How the heck did you get it to do that??" is one of the questions I heard all the time when I was a tester. I took great pride in explaining just how I took all those tangents to get the product to do that. I didn't think my brain thought *that* differently.

Testers are curious. Testers who ask, "why" or "how is this supposed to work" or "how can I break this beast," are more likely to find problems than testers who don't. I want to know how the architecture of the product works. If I understand when the internal structures change, I can create tests to make sure they change, forwards and backwards. One of the marks of a great tester is to be able to create idempotent tests, and I can't do that if I don't know the guts of the product. I want to know.

Testers are observant. Testers who notice patterns of product behavior, or the lack of those behaviors will discover product problems.

I read this a long time ago: http://daringfireball.net/2002/09/welcome_indeed.html and it still resonates with me. Observant testers notice the small differences that make or break a user's experience.

Tester Preferences

In my experience, great testers share these preferences.

Testers are able to plan testing. Not every tester needs to be able to plan everything, but a good tester will be able to plan some of their testing. I'm a huge fan of rolling wave planning. I want to plan enough to be able to do a little testing, report on it, use the results to replan and continue. If I can only do seat-of-the-pants testing and not plan enough to use the results of my testing, I'm not adaptable enough to help my project team.

Testers are adaptable. Testers need to be able to replan or throw out the plan when the plan isn't working, or test a different part of the software, or test a different way. It doesn't matter if you're on a traditional team or an agile team. Every project encounters some sort of issue or problem at one time or another. You want to be adaptable with your testing as well as with your tests. Rigidity in testing is not helpful.



Tester Non-Technical Skills

In my experience, great testers share these non-technical skills.

Testers see disparate events and connect them to develop new and better tests. Have you ever seen something happen in the system over there and something else funky occur over here and then you say, “Aha,” and you try something else, and you bring the system to its knees? That’s exactly what I mean. You can do this with data or intuition; I don’t care. I’m the intuitive type. I bet some of you are the data-driven type.

Here’s an example I saw a few years back. The middleware layer of a transaction-processing system had an intermittent problem. It was data-dependent. We all know how difficult those problems are to find and debug. The testers were testing by feature, and the features separated the problem. One of the testers suspected the defect was data-dependent, and wrote a note on the wiki that she had tried these four combinations. She suggested her two colleagues try another six combinations each. They did, and they successfully crashed the system. They were thrilled. The developers? Not so much. Until the developers realized they had a *reliable* way to duplicate an intermittent problem. Then the developers realized how valuable the tester’s insight was.

Testers write a good defect reports. We talk about “great communication skills” all the time in our open positions. But we really put communication skills to the test when we write defect reports. We have to describe the problem—without blaming anyone. We want to include a short reproducible method to recreate the problem.

Testers choose which defects to champion. If you have a large number of open defects in your projects, you are not going to get to release with all of them fixed. So, which ones of them will you champion? Even if you have release criteria, which ones of them will slip past the release criteria? Great testers take a pragmatic approach to this question and remember that testing is about providing information, not about providing excellence or goodness.

Testers champion a defect. And, when you have defects that you must not release with, because they do not pass the release criteria, you champion the defect. That means you have negotiation and influence skills. You have to be able to influence people across the organization, not just in development, but also in marketing, sales, and maybe in finance or other organizations. That takes skill.

Your Culture Might Need More or Less

Is there some other quality, preference or non-technical skill that makes a difference in your culture? There might be. That’s because your great testers have to work in your culture.

Think about what’s most important to you in a tester. Remember that cultural fit will trump every skill, every single day of the week.

Stay tuned for my next column, where I’ll help you think about technical skills.



Johanna Rothman is the author of *Hiring Geeks That Fit*, <https://leanpub.com/hiringgeeks>.

See her other books at <http://www.jrothman.com/books/>.

She writes an email newsletter, the Pragmatic Manager,

<http://www.jrothman.com/pragmaticmanager/>

Back To Index



A man in a grey suit and blue tie is drinking from a white mug. He has a surprised or stressed expression. To his left is a large, messy stack of yellowed papers. To his right is a large, striped sock. The background is white.

Taking a break?

a click here
will take you there

Dealing with Stress



Stress failures and bug advocacy - looking at stress tests from a value perspective

by Issachar Hazan

Stress is part of your test strategy. You use it as a tool to test your product and find bugs. This is one of the “non-functional” test categories you run. Did you devote the time to think about what is actually being tested by your stress tests?

You may continue to test without answering this question, but when the time comes for bug advocacy, you have to defend your test strategy and findings, and this may force you to search for an answer.

I would like to share with you some “Stories from the trenches” - from my own experience, and use them to base some categorization of stress:

- 1) A stress test revealed disconnection in the communication with an embedded component which transfers its network traffic from its own driver to the host computer Operating System driver whenever the host computer restarts. This was a complex system bug that after a partial fix improved to a “reasonable” failure rate of 1 out of 30 restarts. Following a discussion, we decide to dismantle the stress to a meaningful use case. In our case this was a network session that included 3 restarts. Our logic was that the end user is not concerned with the overall failure rate, but is worried of the chances of his single session to fail. We used a tool that was able to run our defined use case from a clean state over and over and got the statistics of the use case failure rate. This lead to the decision to fix the defect.
- 2) A Software service which runs on a computer system and handles events collapsed after a stress of such events due to wasteful handling of system resources. From a value perspective, this issue is related to the time factor as we can predict that the collapse will happen over the course of time on a system with normal phase of events.
- 3) Data stress that caused a failure of a network connection to an embedded device. Investigation showed that the failure happened starting from a certain amount of data which was much more than the typical usage of the system. Since it was a complex fix and we were at a late stage of the project, the fix was deferred to the next release of the product.

We can categorize the 3 examples to different categories of stress tests:

- 1) Statistical failure - Stress increases the chances of the appearance of a sporadic defect since it executes a flow a lot of times
- 2) Run stability tests in a shorter time - the stress speeds up the time factor - failure reveals in a short time a defect that a system which runs in normal conditions (amount of data, number of simultaneous actions, etc.) will experience after a longer run. A common example of such a failure is a memory leak found using the stress setup.
- 3) Load (sometimes defined as a category by itself) - when we test how our system scales with multiple calls, large amount of data or both. Here, the failure reveals a point when the system fails to handle the load.
- 4) Any combination of 1, 2 or 3.



Issi Hazan-Fuchs has been testing Software, Drivers and Firmware for more than 12 years in the FTL - the Functional Testing Lab of the Intel® design center, in Jerusalem, Israel.

Issi has lead a variety of test teams and projects, as well as promoting testing methodologies and process activities across the testing department.

Issi loves to think and discuss Testing matters. He is an expert member of the Israeli QA forum in "Tapuz".

He publishes his ideas, which are not always standard ones, in his blog:
<http://testeminset.blogspot.com>

In a utopic scenario, when a stress related defect is reported, it follows the path of debug, root cause and fix. But in many cases, we will need our bug advocacy skills in order to convince our stakeholders of the need to fix the defect.

A typical bug discussion can start like this:

Developer: "Stress of 4½ hours and 5MB data files is not a normal usage of our system. A typical use case takes 15 minutes and a smaller amount of data. We should reject this bug."

This point in the discussion can reveal whether you did your homework or not. To decide that the failure is from the 1st classification - statistical, we need to decompose the stress to a meaningful use case and run it over and over while bringing the system to a clean state between the each use case. Automation can be a big help here. If we succeed in reproducing the failure under such conditions, our report will transform from a stress failure report to a use case failure report with reproduction rate. When we have a sufficient statistical sample, the impact is clear.

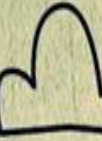
Pinpointing whether the failure is related to time or to load is more complex, as we need to "play" with both factors in order to reach a conclusion about the amount of time, load or both that is needed in order to cause the system to reach a failure point. The awareness of the possible options is an important tool in bug advocacy. For example, it can enhance stakeholder's perspective when you are able to say that "we're not sure yet, but it is possible that we will see the failure in normal conditions after a long period of time." Doing complete research before reporting the stress failure can consume lot of resources and time, so I don't suggest delaying the report till the tester has all of the answers. Many times, we can reach faster and better conclusions about the failure from a focused code review or a debug log analysis. I would like to suggest the following: learn to classify your stress failures. When you see and report a stress failure, treat it as a start of the classification and investigation. While sometimes the report will be enough to call for a bug fix, many times it will serve as a call for investigation. During the investigation - make clear to stakeholders what you already know and what you don't know yet. Make sure that new findings are updated in the bug and don't be afraid to change the title to reflect it.

There is much more to learn than the basics I summarized in this post. Learning more about stress in general and about your specific system, can help you classify and investigate your stress failures and no less important - plan your stress tests better.





are you one of those
#smart testers who
know d taste of #real
testing magazine...?



then you must be telling your friends about ..



Tea-time with Testers

Don't you ? 😊



Tea-time with Testers !

first choice of every #smart tester !



testing intelligence

- its all about becoming an intelligent tester



an exclusive series by **Joel Montvelisky**

Of Testers & Soldiers...

If you google the term "Military Intelligence", among the first results you will find the following **Creed of the (US) Military Intelligence Corps**:

*I am a Soldier first, but an intelligence professional second to none.
With pride in my heritage, but focused on the future,
Performing the first task of an Army:
To find, know, and never lose the enemy.
With a sense of urgency and of tenacity, professional and physical fitness,
and above all, INTEGRITY, for in truth lies victory.
Always at silent war, while ready for a shooting war,
The silent warrior of the ARMY team.*

I couldn't find the source to quote it, but this sounds close enough to what I know about Intelligence Officers from some friends with vast military experience.

Why am I writing this about "soldiers"?

The answer is simple.

A short while ago [Jerry Weinberg](#) commented to one of my posts (published in [Tea Time with Testers](#)) that he was looking forward to my explanation on why I think the work of testers is in many ways similar to the job of Military Intelligence Officers.

I believe he was referring to the following quote from my article "[To Protect and Serve](#)":
"... In many cases we are serving as (Military) Intelligence Officers to our Organizations, helping to make the most complex and challenging strategic and tactical decisions."

Some of you may already know that [Jerry](#) is high in my list of admired testers, so I would not even dream of not answering his questions. So I decided it was time to take upon this topic with a post of its own.

Back to basics: what is the role of the QA Tester?

In the past I wrote my definition of QA (or Testing) Intelligence as follows:

***To provide concrete, relevant and timely information
captured from multiple data sources and using many disciplines
to help our stakeholders make their tactical and strategic decisions.***

This definition is composed of 3 parts:

- (1) We provide the right information
- (2) We gather the information from various sources
- (3) The aim of this information is to help make the correct decisions

Mental Exercise:

Before we move forward I would like you to perform the following exercise. I promise it won't hurt, and it will take you less than 4 minutes to complete it.

PART 1:

Read the definition of QA Intelligence above and think about your current testing team.

- Does the definition help to define the objectives of your team?
- In a high level, does it help you to accurately prioritize your work and to explain to others in your team and outside of it what it is that you are achieving?

PART 2:

Now, think about a Military Intelligence Officer working alongside the Top Generals of an army, and once again read the definition of QA Intelligence but think about the Intelligence Officer's work.

- Does the definition help to define the objectives of the Intelligence Officer?
- Does it help to prioritize his work and explain to other members of his team what are his responsibilities and tasks?

Conclusion:

In many ways the objective of the QA Tester and of the Intelligence Officer are very similar.

Each of us in his or her own contexts, are tasked with providing information that will help our superiors and the rest of the team/unit to do their work better and to make the correct decisions.

We are in the business of Information

Just like the intelligence officer is not tasked with fighting the enemy in hand-to-hand combat, as testers we are not tasked with writing the code that will be delivered to the end users. Our job is to provide information support to the people who are making the strategic and tactical decision as well as those in the fighting and coding lines.

This doesn't mean that we don't have an intricate and highly technical job.

Many times our jobs are even more technical than "only doing the coding", because we need to think about and simulate strange but realistic scenarios where customers will be using our product in unforeseen ways, and that way seek out the issues that may be hiding under these extreme conditions.

Have you ever seen testers using "counter-bug-intelligence" tactics exemplified by the phrase: "if I was a bug, where would I be hiding..."? I know I have!

Now seriously, one of the most complex jobs of QA Engineers is to make sure we are providing the correct information. By correct I don't (only) mean the right data from our test runs, but the actual information derived from processing all our data points and putting together an image that is both accurate and informative.

Not only that, but we need to work with incomplete information, making assumptions and explaining them as risks of things that may or may not happen. Does it sound like guessing where the enemy is hiding and how they will behave?

In the end of the day our stakeholders don't have the time to go over all the results and assumptions. They expect us to do this processing for them. All they want to receive are the synthesized pros and cons, described as simply as possible, to help them make the right decision quickly.

We gather information from multiple sources of raw data

Another thing that connects between testers and military intelligence officers is that we work with a large number of data sources and types.

Just as we need to run functional tests, API tests, Load tests, etc., intelligence officers need to gather data from satellites, personal observations, spies, etc.

For us is not only about bugs and tests but also about statistics of usage and different types of user behavior as well as technological changes and different types of platforms and conditions where our products may be used. In a similar way, for them is not only about the enemy and their weapons, but about the general population in specific areas and the political, economical and even ethnical connections between different factions of a war.

Both we and they need to provide concrete, concise and timely readings of all this information, presenting the current status of affairs and an appraisal of the future based on a limited number of assumptions.

Integrity and truth

Finally, I couldn't help but notice something that was written in the Creed and connect it to words mentioned both by Jerry Weinberg and James Bach in their answers to my [5 Testing Questions](#).

The creed says:

"...and above all, INTEGRITY, for in truth lies victory."

When I read this, I realized it was similar to what both Jerry and James answered to some of my questions.

Like when Jerry answered that the most important piece of advice for a tester would be to:
"Never lie..."

And when James wrote that among another number of traits a tester should always have:
"...a strong sense of ethics."

What do you think?

Are there other similarities between QA Testers and Military Intelligence officers? Do find another profession where with which we share many of the same traits and challenges? Please let us know by writing us!

[Back To Index](#)




Joel Montvelisky is a tester and test manager with over 14 years of experience in the field.

He's worked in companies ranging from small Internet Start-Ups and all the way to large multinational corporations, including Mercury Interactive (currently HP Software) where he managed the QA for TestDirector/Quality Center, QTP, WinRunner, and additional products in the Testing Area.

Today Joel is the Solution and Methodology Architect at [PractiTest](#), a new Lightweight Enterprise Test Management Platform.

He also imparts short training and consulting sessions, and is one of the chief editors of ThinkTesting - a Hebrew Testing Magazine.

Joel publishes a blog under <http://qablog.practitest.com> and regularly tweets as [joelmonte](#)



Call for Articles !

Have you got something to say?

yes, we are listening you...!!!

"Tea-time with Testers" firmly believes that one of the best ways to improve upon software testing is to listen to the lessons learned by others and their experiences too.

So, if you have an interesting story that you'd like to share with the world, contact us at teatimewithtesters@gmail.com.

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

Click [HERE](#) to read our Article Submission FAQs !

T ' Talks



T. Ashok exclusively on software testing

Too many conditions!



The coming of New Year is always interesting. It is the time of the year when new resolutions are made. The word 'resolve' indicates grit, something we strongly wish to comply with.

What does this have to do with software testing/quality? As a software developer, at the start, we make a resolution to deliver great quality software. In real life the New Year resolutions are quietly forgotten as days/weeks pass by. But not in the case of software. It is necessary to meet this and requires effort from developer and tester.

What does it take to accomplish this? (I.e. fulfill the

resolution) It requires one to comply with certain conditions and ensure that they are not violated, thereby exhibiting the new desired behaviours. Non violation of the identified conditions is very necessary to demonstrate 'grit' and thereby meet the resolution.

To deliver great quality software, it requires identification of the various conditions and ensures that:

- (1) Behaviours are as desired when the conditions are met and
- (2) Unexpected behaviour is not exhibited when conditions are violated. Sounds familiar? Of course yes! Test scenarios are really combinations of conditions.

Let's examine the conditions in detail... There are a variety of conditions- they pertain to data, interface, (internal) structure, functional behaviour, (external) environment, resource consumption, linkages to other systems, and other conditions to the system attributes (non-functional aspects).

Enumerating this ...

- 1. Data related conditions: Data types, boundaries, value conditions
- 2. Data interface conditions: Mandates, order, dependency, presentation
- 3. Structural conditions: Linkages, resource use policy, timing, concurrency
- 4. Behavioural conditions: That which governs the functionality, the business logic.
- 5. Flow conditions: The larger behaviour, business logic of end-to-end flow.
- 6. Environment related: Messing up or being messed up by the external environment
- 7. Attribute related: Load conditions, performance conditions, security conditions etc
- 8. Linkages to other systems: Deployment conditions

Ultimately testing is about assessing that behaviour is as desired when all the conditions are combined. Now we have 'Too many conditions'! Now meaningfully pare down the complexity by partitioning. In HBT (Hypothesis Based Testing), this is accomplished by setting up Nine Quality Levels, where each level focuses on certain conditions and their combination. Note the EIGHT sets of conditions that were described earlier map to the Quality Level 1 through 8.

Having partitioned thus, it definitely becomes much easier to combine a smaller set of conditions at each level and ensure compliance and non-violation. Thus chances of meeting the resolution are much higher.

So when we test software, appreciate that all we are doing to is check the 'compliance to' and the non-violation of' combinations of conditions. And to ensure that we are clear if what we want to do, partition these into Quality levels, where only a smaller subset of conditions needs to be combined. So test scenarios are generated at each quality level, and these are complete, smaller and manageable!

On a personal note, my ne year resolution is to be a super randonneur and do a 1000km brevet. So many conditions need to be met to accomplish this - endurance, mental toughness, sleep management,

climbing, and environment resilience. Instead of attempting to combine all at one ago, I have partitioned these and combined a smaller set of conditions and have met with success in the first month.

So what is your New Year resolution? Do not give up or forget! Identify the 'Too many conditions' and break it down and comply. All the very best.

Au revoir.

Level	Objective	Conditions
L9	End user value	User related
L8	Clean deployment	Installation test
L7	Attributes met	Attribute related
L6	Environment cleanliness	Environment related
L5	Flow correctness	Flow behaviour related
L4	Behavioural correctness	Behaviour related
L3	Structural integrity	Structure related
L2	Interface cleanliness	Data interface related
L1	Input cleanliness	Data related

Copyright 2013. STAG Software Private Limited.



T Ashok is the Founder & CEO of STAG Software Private Limited.

Passionate about excellence, his mission is to invent technologies to deliver "clean software".

He can be reached at ash@stagsoftware.com



[Back To Index](#)



OUR PARTNERS

Quality Testing



Quality Testing is a leading social network and resource center for Software Testing Community in the world, since April 2008. QT provides a simple web platform which addresses all the necessities of today's Software Quality beginners, professionals, experts and a diversified portal powered by Forums, Blogs, Groups, Job Search, Videos, Events, News, and Photos.

Quality Testing also provides daily Polls and sample tests for certification exams, to make tester to think, practice and get appropriate aid.



Mobile QA Zone

Mobile QA Zone is a first professional Network exclusively for Mobile and Tablets apps testing.

Looking at the scope and future of mobile apps, Mobiles, Smartphones and even Tablets, Mobile QA Zone has been emerging as a Next generation software testing community for all QA Professionals. The community focuses on testing of mobile apps on Android, iPhone, RIM (Blackberry), BREW, Symbian and other mobile platforms.

On Mobile QA Zone you can share your knowledge via blog posts, Forums, Groups, Videos, Notes and so on.



Testing PUZZLES

by Sebi



Claim your **Smart Tester of The Month** Award. Send us an answer for the Puzzle and Crossword below b4 2nd March 2013 & grab your Title.

Send -> teatimewithtesters@gmail.com with
Subject: Testing Puzzle

“The Bug Bounty”

This exercise is intended to familiarize testers with security bug bounties. Because many testers don't know how to start in this field (I mean security bug bounties, not really hacking or security), I think the best way to see how it's done is to look at examples.

So one of the security bug bounties out there is Facebook <https://www.facebook.com/whitehat/bounty/>

The mission for this month is to **list as many blog posts with valid proof of concepts of an already rewarded bug by Facebook.**

Valid example: <http://www.nirgoldshlager.com/2013/01/another-stored-xss-in-facebookcom.html>

[Back To Index](#)



Biography



Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years. He is currently hosting European Weekend Testing.

He considers himself a context-driven follower and he is a fan of exploratory testing.

He tweets as @testalways.

You can find some interactive testing puzzles on his website www.testalways.com



TESTING CROSSWORD



1		2			3		4
5				6			7
				8		9	
10							
			11				

Horizontal:

1. It is a tool used for stress testing (8)
5. Any event occurring during testing that requires investigation (8)
8. It is testing in which all paths in the program source code are tested at least once (4)
10. It is an integrated Development Testing solution for automating a broad range of practices proven to improve development team productivity and software quality tool (5)
11. It is the initial testing process exercised to check whether the software under test is ready / stable for further testing (5)

Vertical:

1. It is a tool to perform repeatable tasks that help managers, architects, developers and testers to test an application against its performance (5)
2. It is a testing confirms that the program recovers from expected or unexpected events without loss of data or functionality (8)
- 3 It is a measure to detect defects before delivery, in short form (3)
4. It is a type of software testing that seeks to uncover software errors by partially retesting a modified program, in short form (2)
6. It is a testing that exercises a feature of a product in full details (5)
7. A document showing the relationship between test requirements and test cases, in short form (2)
9. The short form of Top Down Testing (3)
10. It is a testing tool and offers an intuitive and extensive solution for automating websites and web applications to execute functional tests and regression tests, in short (3)

Answers for last month's puzzle:

~ Magic Numbers ~

$$1324 * 1234 = 1645732$$

$$2134 * 2143 = 4573162$$

Link to the answer-sheet by **Sharaniya**



*We appreciate that you
"LIKE" US!*



Join us on Facebook.

You are just a CLICK AWAY



Every Tester

who reads Tea-time with Testers,

**Recommends it to friends and
colleagues .**

What About You ?

Our Testimonials



Dear Tea-time with Tester team,

I am Ishwari Salunkhe (web developer). I just read your issue of the month January 2013. I am not a tester but liked you issue.

I read "Eyes Wide Open" editorial by Mr. Lalitkumar Bhamare. It explains a real goal of one's life and what you shall always follow in your life.

Instead of money you pursue your goal, your passion and that's what makes difference.

I never knew that puzzles are mathematical. I also loved the puzzle section in your magazine and look forward to solve it each month.

Thanks for your efforts and all the best for bright future.

- Ishwari Salunkhe
(Adelaide, South Australia)

in ne>xt issue

articles by -

A close-up photograph of a silver-colored metal tag with rounded ends, hanging from a chain. The tag has the words "IT'S ALWAYS TEA—TIME" engraved on it in a bold, sans-serif font. Below the tag, a portion of a metal tea timer is visible, featuring a circular dial and a small loop. The background is a dark, textured surface.

IT'S
ALWAYS
TEA—TIME

Jerry Weinberg

James Bach

Johanna Rothman

T Ashok

Joel Montvelisky

Bernice Ruhland

Ben Kelly

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Contribution and Guidance:

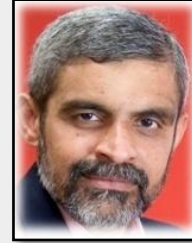
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Image Credits- Olive Cotton

Core Team:

Anurag Khode (Nagpur, India)

Dr.Meeta Prakash (Bangalore, India)



Anurag



Dr. Meeta Prakash

Testing Puzzle & Online Collaboration:

Eusebiu Blindu (Brno , Czech Republic)

Shweta Daiv (Mumbai, India)



Eusebiu



Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)



Kiran Kumar



Chris



Romil

*// Karmanye vadhikaraste ma phaleshu kadachina |
Karmaphalehtur bhurma te sangostvakarmani //*

To get **FREE** copy ,
Subscribe to our group at



Join our community on



Follow us on



www.teatimewithtesters.com

