

THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

NOVEMBER 2012 | YEAR 2 ISSUE X

Jerry Weinberg

Intelligence, or Problem-Solving Ability

Ben Kelly

The Testing Dead – part 2

T Ashok

How do You Solve A Problem?

Martin Jansson & Greger Nolmark

The Best, and Most Humble, Test Team in the World

Joel Montvelisky

To Protect and Serve

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) 9960556841

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.***

'If...' – for Software Testers

I have a question for you.

“Whenever you feel low, what is that thing which sets your mood, motivates you and makes you feel alive again?”

For me, it is good music, novels and one poem which is close to my heart. If you ask me which poem is it? My answer would be **'If...' by Rudyard Kipling**. This poem shows us the way to deal with hard times and situations around us.

On similar lines, most of the times we (testers) come across situations which we can't control and people that we can hardly change. Shall we give up and change our path at such times? Of course not!

When I was reading this poem last time, a funny thought sprigged in my mind. “If this poem would have been written for testers, how would it have looked like?”

I replaced some words in original poem and here is what I came up with. Hope you'll enjoy it 😊

'If...' – for Software Testers*

If you can keep your head when *stakeholders* around you
Are losing theirs and blaming it on you,
If you can trust yourself when all *developers* doubt you,
But make allowance for their doubting too;

If you can *question* and not be tired by *questioning*,
Or being lied about, don't deal in lies,
Or being *neglected*, don't give way to *neglecting*,
And yet don't look too good, nor talk too wise:

If you can think *beyond the specifications* - and not make *just FSD* your master;
If you can *find bugs* - and not make *just bug-finding* your aim;
If you can meet with very *good product* and a *broken code*
And treat those two impostors just the same;

If you can bear the *rejection of bug you have raised*
Misunderstood by Devs to keep the count low,
Or see the *analysis you gave your time to mistaken*
And stoop and build it up with *facts that flow:*

If you can make one heap of all your *test cases*
And risk it on one turn of *requirement change,*
And lose, and start again at your *beginnings*
And never breathe a word about your *rework;*

If you can talk with *team* and keep your *virtue,*
Or walk with *thought leaders-* nor lose the *common touch,*
If neither *developers* nor *fellow testers* can hurt you,
If all *stakeholders* count with you, but none too much;

If you can fill the unforgiving minute
With sixty seconds' worth of *great test ideas -*
Yours is the *product* and everything that's in it,
And - which is more - you'll be a *Good Tester my friend!*

Enjoy another nice issue of Tea-time with Testers!

Yours Sincerely,



- **Lalitkumar Bhamare**
editor@teatimewithtesters.com



QuickLook



Testing Puzzles
by Sebi



Crossword
by



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

The Rise of Zombie Epidemic – 19

In the School of Testing

The Best and Most Humble Test Team in the World – 26

To Protect and Serve -32

T ' Talks

How do You Solve A Problem? – 36

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

The Global Software Testing Services Market to Grow At A CAGR Of 5.16 Percent

By Business Wire

Research and Markets has announced the addition of the "Global Software Testing Services Market 2011-2015" report to their offering.

One of the key factors contributing to this market growth is the reduction in operational time and cost. The Global Software Testing Services market has also been witnessing a shifting focus toward cloud-based testing. Moreover, the shortage of skilled labor could pose a challenge to the growth of this market.

Key vendors dominating this market space include Accenture plc, Cognizant Technology Solutions Corp., IBM Corp., and Wipro Ltd.

Other vendors mentioned in the report: Cap Gemini SA, HP Co., Infosys Ltd., TCS Ltd., Micro Focus International plc., HCL Technologies Ltd., AppLabs Technologies Pvt. Ltd., Logica plc., Software Quality Systems AG, Thinksoft Global Services Ltd., Tech Mahindra Ltd., and Hexaware Technologies Ltd.

Commenting on the report, an analyst from TechNavio's IT Services team said: "The Global Pure Play Software Testing Services market is witnessing a trend of vendors offering cloud-based testing solutions. By moving IT infrastructure to the cloud, organizations reduce the capital investment as well as the cost of maintenance, security, and infrastructure. Furthermore, with the testing taking place over the cloud, testers can work independently to decrease the testing time by accessing the tested software on the cloud even in the product development stage."

According to the report, one of the major drivers in this market is the reduction in operational time and cost. As a result of the intensifying competition in the Global Software Testing Services market, companies are increasingly focusing on product development and R&D to develop a superior-quality product. As a result, companies are increasingly hiring independent testing vendors that specialize in testing services and possess the necessary tools required for testing.

Further, the report discusses that one of the major concerns for the market is the shortage of skilled labor.

For more information visit http://www.researchandmarkets.com/research/mw7xgt/global_software

Original News Report - <http://www.sys-con.com/node/2437229>

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 **97 countries** in the world that read and religiously follow "Tea-time with Testers"?

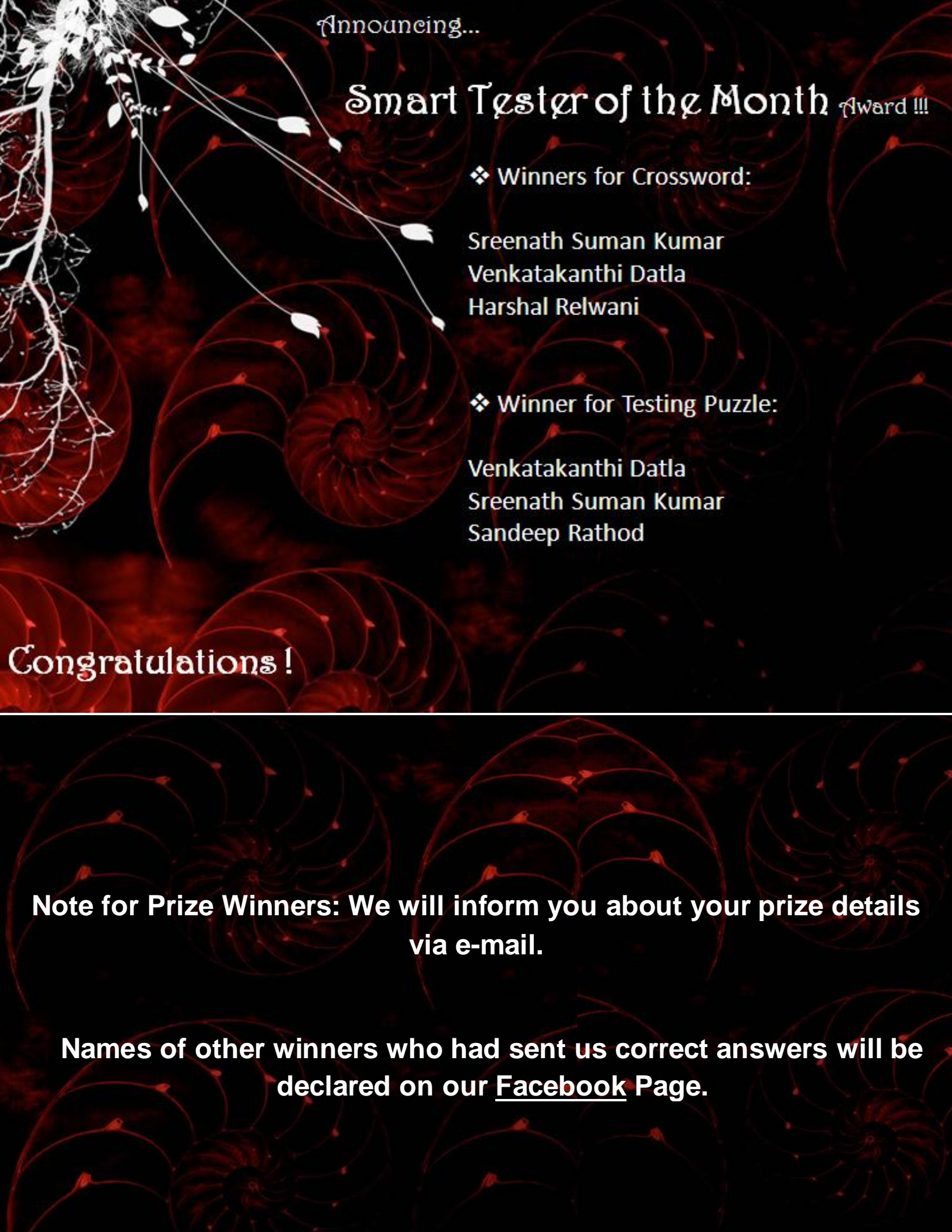
How about reaching industry thought leaders, intelligent managers and decision makers of organizations?

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 !!!

❖ Winners for Crossword:

Sreenath Suman Kumar
Venkatakanthi Datla
Harshal Relwani

❖ Winner for Testing Puzzle:

Venkatakanthi Datla
Sreenath Suman Kumar
Sandeep Rathod

Congratulations!

Note for Prize Winners: We will inform you about your prize details via e-mail.

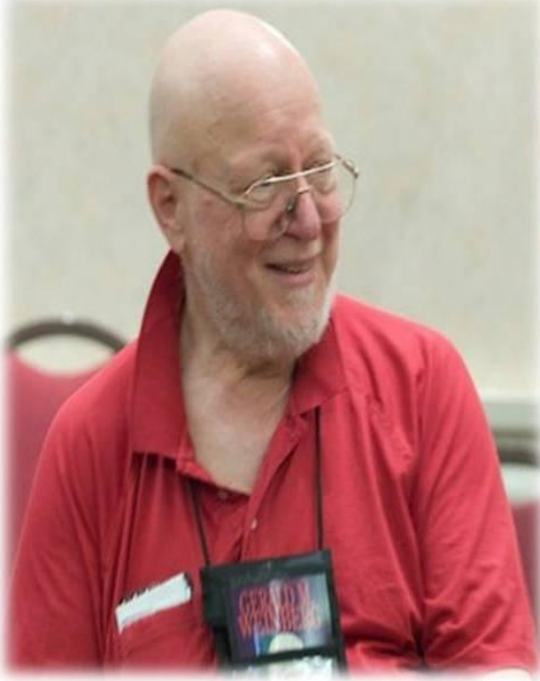
Names of other winners who had sent us correct answers will be declared on our Facebook Page.

Don't miss the action...



join us on FACEBOOK !

Tea & Testing



with

Jerry Weinberg

Intelligence, or Problem-Solving Ability (Part 1)

Working with programmers, is working with people possessing above average in intelligence. Although no formal study has been reported on the subject, we could make a fair guess that the average programmer's IQ is well above the average even of college graduates, and that the more successful programmers, by and large, have an even higher average IQ. Not, of course, that intelligence is all there is to the matter—we have long since disposed of that fallacy. But since above average intelligence is something most programmers possess, we are sure to understand programming better if we look at it in the context of how programming work is affected by intelligence, whatever that may be.

PSYCHOLOGICAL SET

For certain types of error location activities, psychological set proves a major impediment. Numerous experiments have confirmed that the eye has a tendency to see what it expects to see. For instance, when looking rapidly over lists of words, a subject may encounter a "word" which is actually not a word at all, such as "dack." The first influence of set can be seen in the fact that the subject sees this nonword as a word, for he finds it among words and thus has a predisposition, or set, to see it as a word. Secondly, if the subject has been told that the words in the list have to do with "animals or birds," he is quite likely to read "dack" as "duck." If, on the other hand, he has been told the words have to do with "means of transportation," he will much more often read it as "deck" or "dock."

Anyone who has ever tried proofreading is aware of this type of set phenomenon, and anyone who has ever tried to locate a mistyped word in a computer program is more than just aware—he is scarred. In such tasks as proofreading, it is hard to measure the difficulty in overcoming the effects of set, for texts sufficiently perfect to serve as standards and sufficiently difficult to simulate actual conditions are difficult to obtain. Not so in computer programs, for the computer provides an automatic testing ground for the efficacy of proofreading.

The testing a computer can provide for this type of error in its own programs has quite a range of subtlety and power. On the simple end of the range lie the tests for unrecognizable syntax, misspelled keywords, and ill-formed constants. The type of cross-checking provided by symbol tables, cross-reference lists, and flow analyses go one step deeper, but are still essentially static. Dynamic checking for uninitialized variables, flow-tracing, and subroutine call-tracing contribute to cleaning up such typographical errors as may have sifted through the other levels. Nevertheless, no automatic system can be guaranteed to locate all such errors, and we may expect certain improvements by attention to psychological facts when the programs are written or languages are designed.

Related to the concept of "set" is the concept of "distance." Not all misreadings are equally likely, regardless of the set of the reader. Thus, "daxk" is less likely to be mistaken for "duck" than was "dack," because the reader will have to make two letter transformations instead of one. In information theory, two "messages"—which may be taken to be strings of bits—have a "distance" that can be obtained by counting the number of bit positions in which they differ. The importance of this measure is that it indicates the number of bits that would have to be changed to transform the one message into the other—as might happen if noise were present in the transmission.

For the symbols of a programming language, just as for the words in a natural language, such a simple measure of "distance" can only be taken as a first approximation to the likelihood that one symbol will not be mistaken for another. For instance, psychological tests have shown a tendency for initial and final letters to be more significant in making distinctions.

Thus, "gucr" would be much less likely to be seen as "duck" than would "daxk," even though they each differ in exactly two letters from "duck." Moreover, each pair of symbols cannot be assumed to be the same distance apart as each other pair.

Although the exact relationship must depend upon the typescript used, letter pairs such as "x" and "k" would seem to be more readily confounded than such pairs as "x" and "o."

One of the first lessons the novice programmer learns is to make careful distinction between his handwritten "zero" and "oh," if someone else is keying his program. This sort of caution existed before computers, when typesetting was done by hand using slugs of movable type. One particularly common error was to pick a letter's mirror image—d for b, and b for d. From this practice, we derived the expression: "Mind your p's and q's!"

Most programmers, unfortunately, never advance much beyond this point in developing habits that will facilitate the conquest of set as a programming hazard. For example, no matter how carefully one writes the zero in the symbol "STOP," it will be mistaken for an "oh" all along the line. The psychological distance between "STOP" and "STOP" is so slight—because of the similarity of the zero and oh, the middle position of the single differing letter, and the set within the symbol which strongly biases us toward the English word—the programmer who habitually makes such poor symbol choices is headed for certain disaster.

No doubt, the rather extensive success of automatic methods of detecting such errors has lulled many programmers into carelessness when choosing symbols. Nonetheless, there will always be some

situations in which the compiler or interpreter cannot make a sensible conclusion that there is an error. In one case, a programmer had used both the symbols "SYSTSTS" and "SYSSTSTS" in the same code and only discovered that one had been substituted for the other after hundreds of hours of erroneous simulations had been run, a book had been published containing these results, and several systems had been misdesigned on the basis of their errors. All this could have been avoided if he had adopted the practice of keeping a minimum distance of two (dissimilar) characters between his symbols, and perhaps ensuring that at least one of these differences was at the beginning or end.

Mnemonic symbols are particularly susceptible of inducing a torpor in the program reader. Mnemonic symbols expose us to misreading for several reasons:

- They tend to make programs seem "sensible" by their satisfaction of our general set toward sense over nonsense.
- They play upon our tendency to believe in the name, rather than the denotation of the name. Who would believe that the symbol "FIVE" denoted a value of 4? But it did, in one case where the programmer had to modify his code and didn't have time to change all references to "FIVE." He did, however, have time to rerun the program—after having taken much time to locate the source of difficulty.
- They tend to give something less than an optimal "distance" pattern. English words, for example, are not random collections of letters. Some patterns such as consonant-vowel-consonant, or consonant-vowelvowel- consonant, tend to occur more frequently. Even worse, there are homographs such as "LEAD" and "LEAD," which might pop up in the same program from two different origins.

Optimal distance is further reduced because of regularities in the way we abbreviate, leading to such ambiguities as "PEND," for a record that is held in pending status, and "PEND," short for "end of part P." We cannot abandon the subject of set errors without a comment on comments. The whole idea of a comment is to prepare the mind of the reader for a proper interpretation of the instruction or statement to which it is appended. If the code to which the comment refers is correct, the comment could be useful in this way; but if it happens to be incorrect, the set which the comment lends will only make it less likely that the error will be detected.

This effect of comments on interpretation of erroneous code can be measured quite nicely in an experiment in which several versions of the same code are produced, one with correct comments, one with one or two incorrect comments, and one with perhaps no comments at all (Okimoto, 1970). For certain types of code, at least, correct interpretation of what the program does can be obtained more reliably and faster without any comments at all. Correct comments, if well constructed, reduce errors when compared with cases in which incorrect or misleading comments are used. Nevertheless, many experienced programmers make a habit of covering all comments when scrutinizing a program listing for errors, thus reducing set which, though helpful to understanding a correct program, only complicates the already impossible job of debugging.

SOME DIMENSIONS OF PROBLEM SOLVING

In psychology, "set" is usually considered part of the study of "perception" rather than part of "intelligence." Yet it should be clear from the preceding section that set phenomena can influence behavior which we would surely label "problem solving."

Of course, even before the question of problem solving comes, the question of problem avoiding. As we saw, numerous techniques exist whereby a programmer can avoid the problems of set altogether in certain situations. Considered in the abstract, a programmer who avoids a problem altogether is more "intelligent" than one who brings it upon himself, whether or not he ultimately "solves" it.

However, abstract ideas about intelligence rarely fall into accord with our beliefs about concrete situations. Lacking any objective measure, we often judge how difficult a program is by how hard a programmer works on it. Using this sort of measure, we can easily fall into believing that the worst programmers are the best—because they work so hard at it. A case in point was a programmer who worked 14 hours a day, seven days a week, for eight weeks to get a small program running in a new installation. For his efforts, his company gave him an award for exceptional service. Shortly thereafter, another programmer (for the first had been promoted to a management position as an additional reward) was given the job of making some changes to this program. He found the program to be such a confusing mess it was easier to rewrite it than to try and modify it.

The rewriting and debugging took exactly one week, working normal hours. Even considering that writing a program for the second time is far easier than writing it the first, the difference is significant. Moreover, the new program ran eight times faster than the old, took half the storage, and was half as many lines of coding. Clearly, the first programmer had been rewarded for making a mountain out of a molehill. The discovery of this misapplication of management largesse then led to a severe drop in morale in this programming group.

Problem-avoiding behavior, then, is intelligent behavior at its highest, although not very intelligent if one is trying to attract the eye of a poorly trained manager. It will always be difficult to appreciate how much trouble we are not having.

Similarly, it will always be difficult to appreciate a really good job of problem solving. Once the problem solution has been shown, it is easy to forget the puzzlement that existed before it was solved. For one thing, one of the most common reasons for problem difficulty is overlooking some factor. Once we have discovered or been told that this factor is significant, working out the solution is trivial. If we present the problem to someone else, we will usually present him with that factor, which immediately solves nine-tenths of the problem for him. He cannot imagine why we had such trouble, and soon we begin to wonder ourselves.

Overlooking a factor in a problem is just one special case of assumptions leading us astray. We assume that a certain factor is not important—probably without even thinking about it in any conscious manner. We are led similarly astray by assuming that a certain factor is important, when it has no significance for the problem at hand. People who spend much time debugging other people's programs soon learn not to listen to explanations before tackling the problem. Such explanations tend to put the listener on the same false track of assumptions that prevented the speaker from finding the bug for himself.

Psychological set, of course, is another form of making assumptions. Although the assumptions may be buried more deeply in this case, they have the same effect on problem solving. Could we not say, then, that the first rule of problem solving is "don't make assumptions"?

We could say that, but we would be precisely wrong. If we are to be successful at solving problems, we must make assumptions. If we really faced each problem as entirely new, it would be impossible to improve our problem-solving performance. The set we have, for example, which enables us to read a misprint as if it were correct, is a most valuable asset— in most situations.

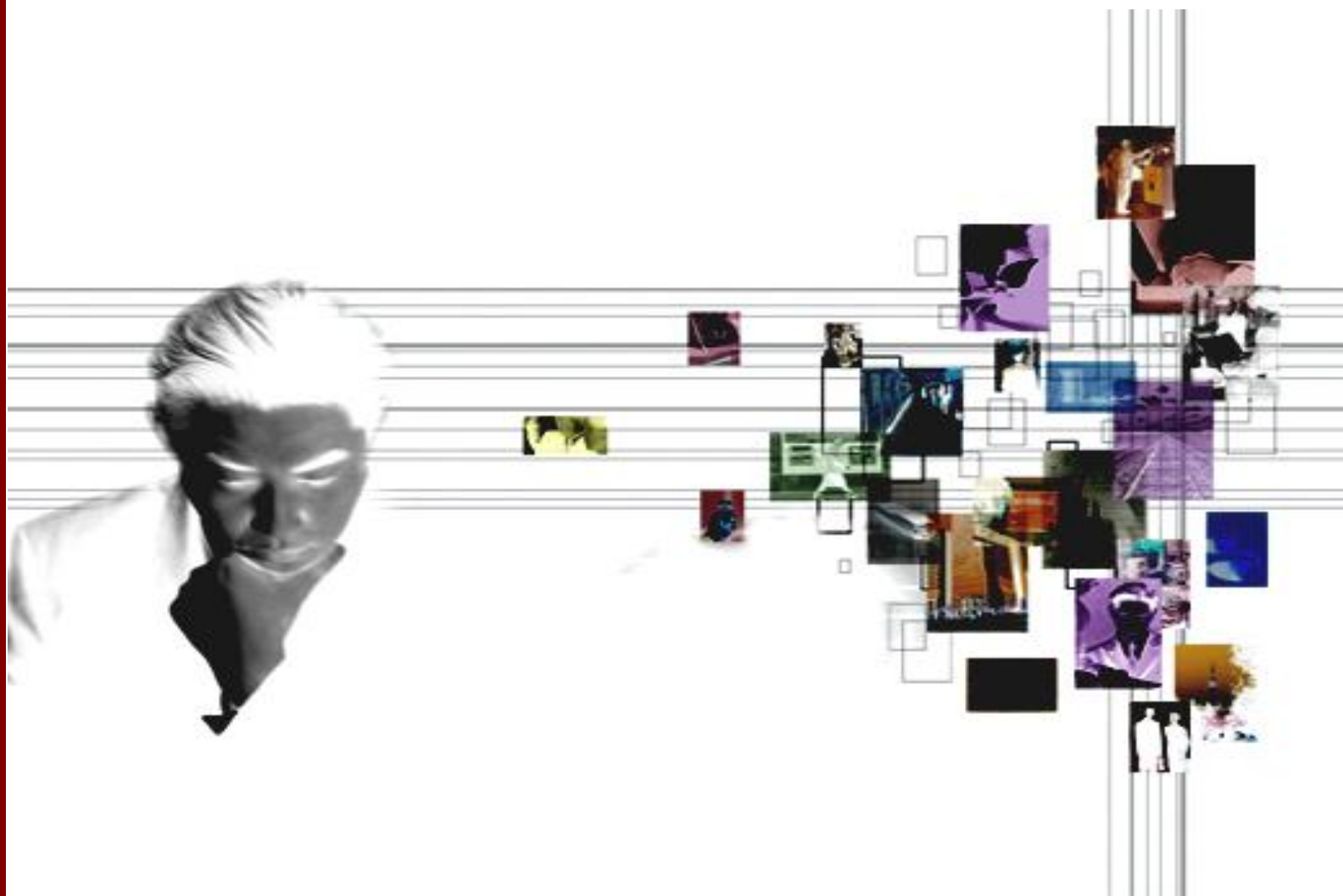
Only when we are proofreading, something few of us spend much time doing, does this particular set cause trouble. Intelligent behavior, then, does not consist in eschewing assumptions, but in being sufficiently flexible to manipulate assumptions as the occasion demands. In other words, being intelligent is not having some magic formula which one can apply to every problem. It is, rather, having a number of "formulas" and not being so much in love with one that it cannot be dropped for another.

Before we leave the topic of adaptability in problem solving for closer examination of some of the "formulas" selectively applied by the successful problem solver, we must lay to rest one more fallacy about intelligence. Intelligence, however it is ultimately defined, is, at best, a statistical concept. We cannot ever hope to measure intelligence by performance on one particular problem, for there are as many non-intelligent reasons for getting the "right" solution as there are intelligent reasons for getting the "wrong" solution. Indeed, the explanations for success given by some programmers bring to mind the story of the village idiot who won the monthly lottery. When asked to explain how he picked the winning number, he said, "Well, my lucky number is seven, and this was the seventh lottery this year, so I multiplied seven times seven and got the winning number—63."

And, when someone tried to tell him that seven times seven was forty-nine, he merely answered with disdain, "Oh, you're just jealous"—which, of course, was true.

to be continued in next issue...

[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*.

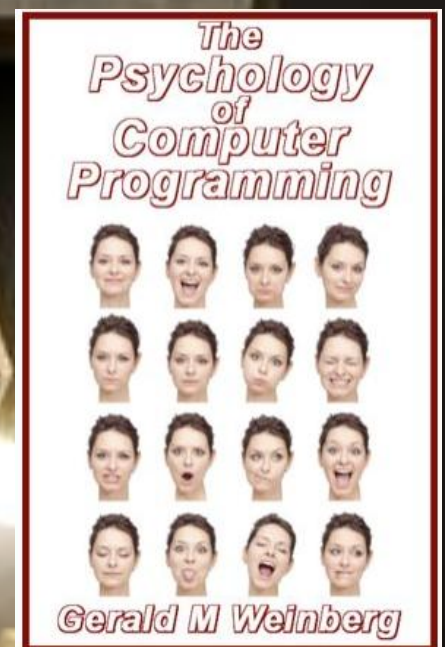
In 1993 he was the Winner of the **J.-D. Warnier Prize for Excellence** in Information Sciences, the 2000 Winner of **The Stevens Award** for Contributions to Software Engineering, and the 2010 **Software Test Professionals first annual Luminary Award**.

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 circular pattern of concentric, slightly raised ridges, resembling a ripple in water or a footprint. The entire scene is framed by a dark blue border.

Speaking Tester's Mind

- straight from the author's desk



In my first article on The Testing Dead, I identified a number of patterns of behavior that I like to call Zombie Testing.

Is this really a problem we need to be concerned about?

I think it is, for a number of reasons.

I think Zombie Testing has the ability to infect an organization. It's a generally less grisly process than your traditional zombie, but the downside is it takes a lot longer to die and it's only slightly less painful.

How does Zombie Testing infect non-testers? I mentioned in a previous post things like arbitrary entry/exit criteria. Have you ever seen programmers changing bug severity or priority (or reassigning them or closing them) to meet these bogus criteria? Ever been in sign-off meetings where project managers argued about which bugs were severity 1 and which were severity 2 and go on to (re)define what they meant?

This one little artifact that says 'no more than 1 severity 1 bug and 5 severity 2 bugs or else your code doesn't get signed off'. It's a sign that zombie testing has taken hold. Anyone with kids will tell you – it doesn't matter if it's number one or number two, you just have to take action before it gets messy.

When Zombie Testers hold themselves up as the quality police, there's a tendency for others to see them that way also. That invites dysfunction like the segregation of testers and programmers – because the dark gods forbid they should unduly influence one another. The testers need to remain "objective". Segregation of testers and programmers is one of those 'won't someone think of the children' arguments. It's a solution in search of a problem that I've yet to ever actually see.

Imagine a straight-laced chaperone at a formal high school dance, insisting that testers and programmers may dance, but must keep at least two feet apart whilst they do so. That might seem very civilized and genteel, but everyone knows the real magic happens when the testers and programmers slip away behind the bike sheds and show each other their notes.

More recently I've heard and read about some programmers calling to do away with testers all together. It's a misguided notion, but I can understand where they're coming from. If your only exposure to testing has been with people that enforce unhelpful rules, have an adversarial attitude, waste your time and otherwise make your life difficult, (whilst adding questionable value) why wouldn't you want to do away with them?

The problem for thinking testers isn't so much that Zombie Testers exist. It's that they're so prevalent that they're seen as the norm by non-testers. We need the people that hire testers, the people that manage testers and the people that testers serve to understand what it means to be a thinking, professional tester.

Moreover, we need to them to understand it in a way that's meaningful to them.

Easier said than done. It's a tough sell.

Can we go to upper management and tell them that quality will improve as a result of our participation?

No.

It may indirectly, but that's not generally something we have direct control over. We don't make design decisions, we don't hire or fire programmers, we don't decide what gets fixed or deferred – we might influence one or more of these things, but the final decision is not ours.

Can we tell them their product will be released bug-free?

No. Finding bugs is part of what we do and while we can test for their presence, we cannot prove their absence. Some less scrupulous companies (who may well have a large stable of test zombies corralled somewhere) might say otherwise, but that's not a claim a tester can make in good conscience.

What then?

The alternative we have is to tell them that we can reveal risks and problems to them much earlier than they might otherwise find out about them, giving them time to take action.

It doesn't sound like a particularly attractive alternative. In my experience, people don't want you to tell them about problems (unless you're also telling them about how you fixed them). They want solutions.

Moreover, many people seem to cling to the broken Taylorist model that software development is mass production. Programmers turn out widgets that come down the conveyor belt. Testers pick up these widgets, compare them to spec and/or known good widgets and if they're within tolerable limits of variance then all is well.

It's an attractive fantasy. It's measurable. It's controllable. The workers can be switched in and out because it's repeatable labor. Unfortunately (for those that believe it), it's complete bullshit.

So how do we put that alternative in a way that is more palatable to an audience that needs to hear such a message, but may not be ready to accept it?

There are no easy answers to that question (that I know of). There's no silver bullet.

In episode three I'll talk about what can be done to help educate our non-testing peers about what software testing is, and what can be done about stemming the flow of zombie testers.

Ben Kelly is a software tester living and working in Tokyo, Japan.

He has done stints in various industries including Internet statistics, insurance and most recently online language learning.

When he's not agitating lively discussion on other people's blogs, he writes sporadically at testjutsu.com and is available on twitter @benjaminkelly.



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 large, rusted metal sculpture in a grassy field with a butte in the background. The sculpture is composed of several large, rounded, interconnected forms, resembling a stylized figure or a complex knot. It is made of a material that has rusted to a deep orange-brown color. The background shows a vast, open landscape with a prominent butte in the distance under a clear sky.

The best, and most humble, test team in the world

by Martin Jansson & Greger Nolmark

Introduction

Why is this important? The view on testing has been terrible for a long time. Many organisations view testing as a necessary evil. Many people have been “demoted” to testers, who now lack ambition, self-confidence in going somewhere else or trying to become great in their current role. Some believe that a mere certificate creates a great tester, but in fact it just hurt the reputation of testers [1]. Some have stopped caring about their work as a tester, stopped their cooperation with non-testers and ignore what value they provide to the organisation [2]. Many testers accept that someone else write them detailed instructions on how to exactly perform their work, a tayloristic view of the (zombie) tester [3], where no knowledge or skill is needed to be able to execute the script. Some call what they do testing [4], but is it something else? Many are uncertain how to get better and do not strive for becoming great [5].

Many of these things stop you from being jelled as a team, from being creative, from having the freedom to do things that fit best in your context or in essence from being great.

Background

After several years of struggle within the QA department, at a former employer, we reformed into something new. Instead of being split between the projects, working differently, with different workload, different ambition and with varied result of testing, we rallied under one flag, unified under a set of goals. Former conflicts and misunderstandings were shaded out by working together with new found respect, but first of all with too much to do.

Some years earlier we had discussed the test team motto and tried to find one that was suitable for us. We had read Brian Marick's Test Team Motto [6] and let that guide part of where we were heading. We were also greatly inspired by the Black Team from Peopleware by Timothy Lister and Tom DeMarco [7]. We were setup in a way so that we served many parts of the organisation (several projects, support, marketing and CEO). After having this setup for a while, we started to grow in pride of our work and our skill. It was then that we started to call ourselves "The best, and most humble, test team in the world!" with some seriousness and a lot of humour. We challenged the rest of the organisation to bring us more things to test, we would report enough bugs to keep the rest of the organisation occupied. This was a clear statement to the rest of the organisation about our ambition, but was this our motto as well? What did it actually mean?

Our line manager for QA did not react well in how we behaved at the time. The company culture nurtured and valued individual success, but our team valued that the whole team succeeded. Was this one of the reasons perhaps? Sometimes we forgot the humility trait when our pride and cockiness got too high. Still, we aimed for being great and I think we were. Looking back at the team members at that time, most have gone separate ways but the journey we took together has certainly made us greater testers.

What is included in the concept of "The best, and most humble, test team in the world!"?

Expectations

To be able to show an ambition and mindset that fits with being able to state the motto "We are the best, and most humble, test team in the world!" we have identified several important aspects to consider in specific roles and in the team as a whole.

Scott Barber has expressed his view of being a context-driven tester [8] that resonates with our thinking and expectations of someone working in the testing domain.

Tester expectations

- You are among many things a service provider to many different stakeholders [9].
- You have a desire to learn new things
- You are humble towards your peers, knowing that things can be different and that anyone can make mistakes, even you as a tester.
- You work with the right things in a fast pace with the right quality
- If something takes a long time, you make sure you are on the right track by asking for feedback
- If you are unsure of priority of things, you ask stakeholders and co-workers
- If you think that something have wrong priority you communicate this
- You resolve conflicts directly or bring in a manager if help is needed
- You spend a bit of time, based on common sense and context, during regular work hours to get better. On your spare time you are free to do what you want
- If you determine that something is meaningless to do, better investigate if it is or if you have misunderstood the intent. You are not allowed to work on meaningless tasks that provide no value.
- You praise in public and criticize in private
- You question given routines and methods
- You are interested in testing and view it as a valid profession and not just something to do “when no other tasks are available”
- Never wait, there is always something to do, whether it is helping a team member or another team finishing a task, working on something that you have postponed for a long time that would feel good to finish or working on something that give value in the long run.
- No tester is perfect or the greatest. Instead it is by adding his or her specific perception or experience to the team that is valuable [10].

Manager, Test Lead or Team Lead expectations

- You train the team in taking on many roles
- You train the team in being backup test lead or team lead
- You gather information and communicate what is important
- You are a tester foremost, not just planner, leader or manager. The same expectations for a tester, applies to you.
- You are active in daily testing tasks if possible. You should identify yourself as a part of the team and should also stay close to the team to be part of the “flow”

Team expectations

- The mindset should be Quality Assistance [11] and dealer in information
- As a team you work on your test team motto or mottos (see [6] and [12]). Examples could be "Replace fear of the unknown with curiosity" [13], "I'll help with anything, but it's you who needs to do the work" [14] or "We are the best, and most humble, test team in the world!"
- The success of the team is more important than that of the individual's. We help each other to succeed
- You should empower and trust your team members. This will yield a higher sense of security and confidence. Trust in each team member's competence
- Everyone in a test team tests, this is our core function. If you do not want to test, then you should not be in a test team or test organisation.
- Market the test team to the organization. Hard to do good work if no one cares what you do or that they are even unaware of your existence.
- The team need to be "fearless" and have the courage to stand up to their opinions, bugs etc. Need to know when to back down.
- Keep an open mind. Do not follow one strict set of rules, unless absolutely necessary due to external reasons. There is no such thing as "We are perfect". In the team you should feel that you can show weakness, it's by making mistakes we learn.
- In many test organisations there are many skeletons in the closet, which need to be cleaned out [11]. The team needs to help clean these out and communicate why that is so.
- Look for diversity of team members. There are many layers in testing, from UI to shell to scripts. Try to get people with many areas of expertise and different backgrounds. [15]
- Previous test experience is not always necessary; other experience may be just as important, fresh eyes etc.
- Be part of recruiting more testers to the team and make the final decision if someone is suitable for your team.
- The team needs to feel they have the freedom to choose their own methods and processes, since they are the test experts
- Visualize what you as a team are working on so that it is easier to dodge tasks from the outside that are non-test-related and mostly distractions
- A clear view on "why do we test". Compare this with Autonomy, Mastery and Purpose (Dan Pink) as motivational factors. We need to have a purpose with our work in order to be motivated and thrive.
- As a team you should work close together, trying different constellations
- Cultivate social interaction within the team. A team with high social cohesion might "jell" and be more productive. A key indicator is whether the team members enjoy spending time together

Conclusion

The aspects above are some things that we have identified, most likely there are several more. Some might only be relevant to us, but there might also be a few that resonate well with your own context. We are humble in knowing that things can be different for testers, their teams and organisations. In our quest for becoming a better, even the best, we keep the above expectations as a foundation for our mindset as a test team and the organisation around it.

References

- [1] <http://thetesteye.com/blog/2011/05/testers-greatest-nemesis/>
- [2] <http://thetesteye.com/blog/2010/11/turning-the-tide-of-bad-testing/>
- [3] <http://lets-test.com/wp-content/uploads/2012/05/LetsTest2012-BenKelly-TheTestingDead.pdf>
- [4] <http://thetesteye.com/blog/2012/03/don%C2%B4t-hustle-my-flag/>
- [5] <http://lets-test.com/wp-content/uploads/2012/05/Soyouthinkyoucantest-Letstest-20120509-Print-version.pdf>
- [6] <http://www.exampler.com/testing-com/writings/purpose-of-testing.htm>
- [7] Peopleware by Timothy Lister and Tom DeMarco
- [8] <http://www.testingreflections.com/node/view/8657>
- [9] <http://www.satisfice.com/blog/archives/652>
- [10] <http://thetesteye.com/blog/2011/02/there-are-no-testers-that-are-the-best/>
- [11] <http://www.kaner.com/pdfs/TheOngoingRevolution.pdf>
- [12] <http://www.exampler.com/testing-com/writings/another-motto.htm>
- [13] <http://blog.softwaretestingclub.com/2010/09/the-software-testing-motto/>
- [14] <http://testers-headache.blogspot.com/2009/10/whats-your-testing-motto.html>
- [15] www.kaner.com/pdfs/JobsRev6.pdf

[Back To Index](#)



Martin Jansson, owner and consultant at Testverkstaden Sverige AB, started his career as tester 1996.

He has tried many professions in product development, but his heart and soul belongs in testing. Martin is one of the founders of www.thetesteye.com which has grown into one of the greatest Swedish blogs on software testing.

Martin is a frequent runner of the testlabs at various test conferences. The last years Martin has assisted clients in evolving their organizations from a traditional one into an agile, where he focuses on what aspects of testing that works in an agile environment.

You can reach him on Twitter @martin_jansson or on martin.jansson@testverkstaden.se



Greger Nolmark, Senior Test Consultant at Adecco IT-Konsult, started working with test in 1999. During the years he has had several different positions in the areas of test and support.

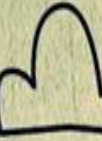
His primary focus has been on testing always keeping an eye on the end user perception.

During recent years he has nurtured an interest in how new and different test techniques can help teams and testers to test better and to show there are more to testing than being certified.

You can reach him on greger.nolmark@adecco.se or Twitter @GregerNolmark.



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**

To Protect and Serve

If you'd need to choose a motto for the testing profession and all the testers worldwide, regardless of the company they work in or applications they test, what would it be?

For me, the choice would be - To Protect and Serve

And not because we are "testing policemen" patrolling the "functional streets" of our AUTs (Applications Under Test, for the non-testers among us) looking for "criminal bugs" and placing them in "bug-tracking-jail" like petty thieves – although the mental picture is kind of funny. But because our responsibility as part of our team and organization is to serve our internal and external customers (e.g. the external end-users; and also the internal developers, product managers, executives, etc) and to protect them from making the wrong decisions about the product we are developing and the process used to develop it.

Protecting Who and From What Exactly...?

Boiling it to the minimum, I believe that as testers we are here to:

(1) Make sure that our teams are developing the right product – with the right features, answering the real needs of our users, without any unwanted issues, etc.

And at the same time:

(2) That our teams are developing the product right - following the process we decided to follow, without wasting unnecessary time, working in a socially and economically efficient way, etc.

We are here to provide visibility into the product and the process, to reduce the time to market uncertainty and to answer the \$25,000 question or whether we are ready to release or not – in the past I've called this QA Intelligence.

And if all this wasn't enough, in many organizations we are also been asked to lead the task of performing Risk Analysis and Management throughout the end-to-end development lifecycle.

Risk Analysis?

Simply put, as a tester you should make sure that if there are any risks that may affect the project from been completed on scope, on time and on budget, then these risks need to be identified, tracked, if possible avoided, and if not then they should be handled correctly.

I will write more about risk management in the context of QA & testing in a future post. But for now I want to add it as another one of the tasks we are doing in order to protect and serve our customers.

Bottom Line... Are We Policemen?

It is true there used to be a stigma of seeing the tester as the bug-policeman some 10 to 15 years ago, and I believe there are some development teams where this might still be the case (let's call these guys cave-men-developers, since they seem to be still leaving in the stone-age of software development).

But Reality seems to evolve, and so in today's development practices, and as we become better professional testers and QA specialist, more and more of the actual tests and bug detection activities are carried on by developers and others members of our organizations.



And we as testers are taking on a different responsibility of guiding and steering the process in the right direction. In many of the cases we are serving as (Military) Intelligence Officers to our Organizations, helping to make the most complex and challenging strategic and tactical decisions.

What do you think?

What motto would you choose for us testers, and why???



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](#)

[Back To Index](#)





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

How do you solve a problem?

Every moment in life is an interesting one, as we encounter new problems that we are challenged to solve. Some of these are ones that we have encountered before and therefore we know the solution, while some are indeed new, and we have to figure out the solution. Now, how do you solve any problem?

Well the easy answer is "Based on experience". You have encountered the problem before, solved it and therefore have to apply the same or a modified solution to the current problem. Sounds familiar? A common question that we encounter - "Do you have the relevant experience in this domain/technology to test this software?" The premise is if I have tested similar systems, I should be able to do a decent job with the current system.

The other answer - "Based on sound logic/technique". I have not solved it before, but I know the algorithm, the technique, the logic to solve it. A great answer, but people are skeptical, unless you are "certified" in the application of it. Techniques are the result of scientific thinking, the application of logic.

Now what do these imply? In the former case, you need to have experience that takes time and is expensive while the latter is cheaper as it can be taught. Are there any other ways? Yes, in some cases, we are taught certain principles, that we apply. Principles are not exact techniques (formula), but are conditions that we use to make choices to solve a problem. For example, a simple principle is that if you are walking eastward and the shadow is behind you implies that it is still morning, while the shadow in the front implies evening. So if a question (I.e. a problem) was posed to you to figure out the time of the day given the shadow position and the direction of walking, applying the principle of shadow gives you the answer.

In some cases, we may not have an exact formula or a clear set of conditions, but a set of directions to choose from based on some information. These are "Guidelines", that identify different situations and suggests what to do in each case. For example a guideline may enable you choose a test technique based on the type of fault you wish to uncover. In the case of complex logic, use code coverage techniques, in case of complex behavioral conditions, use decision table.

Let us attempt to create picture of this...

Problem solving based on experience is based on the "skill of an individual" while the one based on the logical/scientific thinking depends on the "strength of the process". Skill-based problem solving can be seen as an 'art/craft' while a logical approach can be deemed one based on science & engineering i.e. scientific principles + process of usage of these principles.

The figure alongside depicts the problem solving approach as being dependent on these two aspects - "People Skill" and "Process Strength".

In the context of scientific approach to problem solving, the approach can be further be divided into three approaches:

- (1) A formula-like approach, algorithm in nature titled "Technique" based
- (2) Decision enabler that outlines key points to make-choices/choose-path titled "Principle"

(3) A broad brush approach that is suggestive of situations and what may be done in these situations is titled "Guideline".

People skill	Process Strength	Problem solving approach	
C	A	Technique	Science & Engineering
B	B	Principle	
A	C	Guideline	
A+	C-	Pure experience	Craftsmanship

Note:

1. People skills are rated as A+ through C to indicate the individual skillfulness needed from : Highly Experienced/Skillful to Least

2. Process strength are rated as A+ through C to indicate how strong the process/technology is : From Very Strong to Least Strong

The ideal approach would be one that is based on "Technique", so that one can this to an individual so that this can be predictably applied rather than solely on one's experience. Note that acquiring the experience does take time, and this cannot be shortened drastically.

The crux of HBT (Hypothesis Based Testing) is based on a set of thinking disciplines that has set of tools based on the three problem solving approaches of Technique, Principle and Guideline.

The fun part of being an engineer is encountering problems and devising solutions. So the next time you solve a problem, identify what the approach was: Did you discover/apply a 'Technique', 'Principle', 'Guideline'. This will greatly help to build and refine your problem solving toolbox. And it is a wonderful feeling to have a great toolbox. A big Swiss Knife to solve all problems.

Well, what is the approach to solve problems that I encounter with my spouse? None of the above! "Accept whatever she says" and the problem is solved!!

Well until next time, have fun. CIAO.

[Back To Index](#)



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





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.

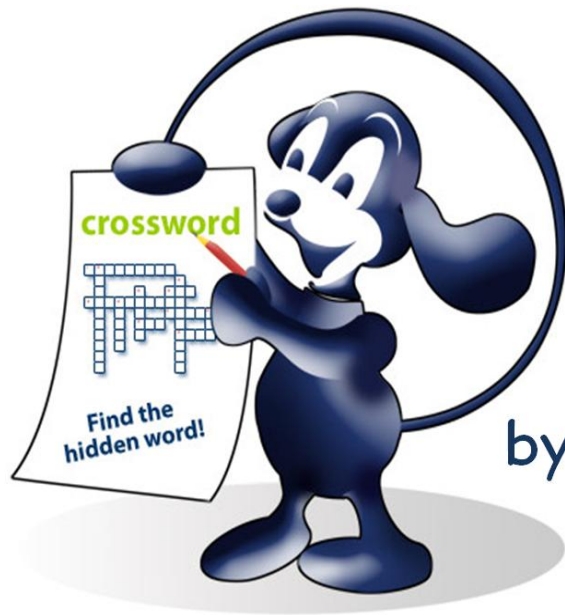
Looking at the scope and future of mobile apps, Mobiles, Smartphones and even Tablets, Mobile QA Zone has been emerging as a Next generation software testing community for all QA Professionals. The community focuses on testing of mobile apps on Android, iPhone, RIM (Blackberry), BREW, Symbian and other 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 bellow b4 20th Dec. 2012 & grab your Title.

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

“Find the next number in the series”

26

72

53

77

71

15

69

46

18

55

83

98

?

[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. A device, computer program, or system that accepts the same inputs and produces the same outputs as a given system. It is called ____ (8)
5. Tool for testing performance and scalability of web services (5)
7. It provides an efficient means for generation, organization, and execution reporting of test cases among projects and by multiple testers and versions, in short (3)
8. The sudden and complete failure of a component (5)
9. It is a certification for Software Test Engineers (4)
11. It is a visual technology to automate and test graphical user interfaces using images (6)

Vertical:

1. It is a Unix automation and testing tool (6)
2. Testing the ease with which users can learn and use a product, in short form (2)
3. After the designing and coding phase in Software development life cycle, the application comes for testing then at the time the application is stated as ____, in short form (3)
4. Confirms that the program recovers from expected or unexpected events without loss of data or functionality. It is called ____ testing (8)
6. Open Source Load Testing solution that is free and cross-platform (6)
7. A Simple Test Driver Generator for Ada Programs (2)
10. Running a system at high load for a prolonged period of time, in short form (2)

Answers for last month's Crossword:

T	E	S	T	C	U	B	E
		E		O			M
B	R	E	A	D	T	H	U
L		T		E			L
E		E			B	V	A
R	E	S	U	L	T		T
B		T		T			O
Y	M	A	N	T	I	S	R

Answer for last Testing Puzzle:

29 Oct 2012 - Monday	29102012=1
11 April 1342 - Wednesday	11041342=3
18 August 4234 - Monday	18084234=1
5 January 1000 - Sunday	05011000=7
14 December 5674 - Friday	14125674=5
7 April 0002 - Monday	07040002=1
31 May 1743 - Friday	31051743=5
20 September 1976 – Monday	20091976=1



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

I have recently started reading TTWT magazine and was really amazed by the content and the way it is presented.

Generally technical magazines are very formal in their presentation which leads to lack of interest after reading it for sometime but TTWT has taken care of this and ensured that the readers are glued to the magazine by presenting the content in a very creative manner. I am not an avid reader but when I read TTWT I just don't feel like stopping until I have completed the entire magazine.

It is really building my knowledge base, improving my analytical and problem-solving skills and keeping me updated about the testing field.

Thanks to the entire family of TTWT magazine for bringing such a wonderful magazine to the testing community.

I wish you all the best for future!!

Suman Chakraborty

Pune

in ne>xt issue

articles by -

A photograph of a metal tag with the text "IT'S ALWAYS TEA—TIME" engraved on it. The tag is attached to a chain. Below the tag is a small metal clock with a circular face and a small tag hanging from it that says "TEA". The background is a dark, textured surface.

IT'S ALWAYS
TEA—TIME

Jerry Weinberg

T Ashok

Joel Montvelisky

Bernice Ruhland

Ben Kelly

Lev Lesokhin

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Contribution and Guidance:

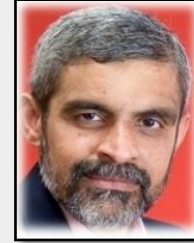
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Image Credits-weipphoto.com

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 kadachna |
Karmaphalehtur bhurma te sangostvakarmani //*

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



Join our community on



Follow us on



www.teatimewithtesters.com

