

Year 1 Issue II

March, 2011

# Tea-time with Testers

THE INTERNATIONAL MONTHLY ON SOFTWARE TESTING

Joel Montvelisky

Testing Intelligence - Test Case Writing

Anuj Magazine

Managing Multiple Passions

Martin Jansson

Turning The Tide of Bad Testing

Gerald Weinberg

Testing without Testing

James Bach

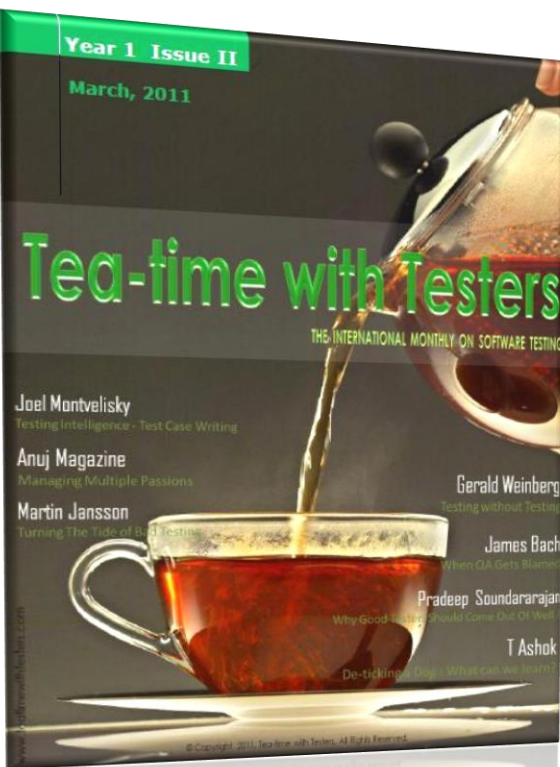
When QA Gets Blamed

Pradeep Soundararajan

Why Good Testers Should Come Out Of Well ?

T Ashok

De-ticking a Dog : What can we learn?



Created and Published by

**Tea-time with Testers.**

Hiranandani, Powai- Mumbai -400076  
Maharashtra, India.

Editorial and Advertising Enquiries:

Email: [teatimewithtesters@gmail.com](mailto:teatimewithtesters@gmail.com)  
Pratik: (+91) 9819013139  
Lalit: (+91) 9960556841

© Copyright 2011. Tea-time with Testers.

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 editors of **Tea-time with Testers** ezine.

## Editorial

Dear Readers,

First of all, I thank you all on behalf of our entire team for making the very 1<sup>st</sup> issue of "Tea-time with Testers" a huge success.

The response we have received is overwhelming and that has certainly boosted our confidence level.

Below verses I have always found very meaningful –

*"Karmanyē vadhīkaraste ma phaleshu kadachna|*

*Karmaphalehtur bhurma te sangostvakarmani ||"*

- Bhagwat Geeta

Meaning:

*"You have a right to perform your prescribed action, but you are not entitled to the fruits of your action. Never consider yourself the cause of the results your activities, and never be associated to not doing your duty."*

How true...! Before launching this ezine, the only thing we had made sure was to give our maximum & honest efforts towards contributing to our software testing community. Going forward we are committed to perform this duty of contributing back to our testing community.

Gone are the days when Testers used to be considered as *helpers* in many organizations. Time has drastically changed now and Testers are being considered as *commodity* in the industry. Needless to mention that, this is the fruit of the endless efforts taken by many legendary testers and their invaluable contribution to the field of software testing.

Today, testers are meeting each other, learning from each other and in a way they are contributing a lot for the betterment of this field. It is worth praising, but the real success will be then *when every tester in the industry will be a Tester by Choice and not by a Tester by Chance*.

Yes...! Miles to go...before we all sleep...!

Come! Let's join hands and make this miracle happen.

Yours Sincerely,

Lalitkumar Bhamare.





# QuickLook

The logo for 'levelfour' features four blue dots of increasing size to the left of the word 'levelfour' in a lowercase, sans-serif font.

4

## Editorial

### What's making News?

### Speaking Tester's Mind

Testing Without Testing – 8

Why Good Testers Should Come Out Of Well? -13

Managing Multiple Passions - 18

### In the School of Testing

Turning The Tide of Bad Testing (Part 1) -23

When QA Gets Blamed - 28

Testing Intelligence: Writing a good test case - 31

The SCRUM Primer (Part 2) – 35

### T' Talks

De-ticking a dog – What can we learn from this? - 42

### Announcements

### Tool Watch

### Our Testimonials

### Family de Tea-time with Testers



8



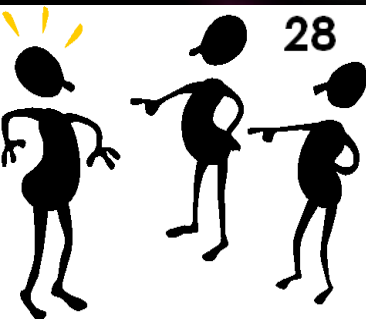
13



18



23



28



31



35



42



48

# What's making News?

- find out the latest happenings in the technology world

## Level Four updates ATM testing software

The updates to Level Four's BRIDGE:

**T**est, Regression Test Manager (RTM) and ATM Developer products enable banks to create and run a greater number of tests more rapidly, therefore maintaining higher levels of ATM uptime for consumers while reducing the time and cost invested for banks.

Ian Kerr, CEO of Level Four, says, "Many banks' ATM test strategies fail to adequately cover all the required scenarios, leaving them vulnerable to costly and unnecessary network downtime. We continually invest into R&D each year to ensure that our test automation solutions retain an industry leading position. This latest product offering is testament to our commitment to deliver the best software and services to our customers, enabling them to focus time and resource on revenue-generating ATM activities."





The new versions of Level Four's products, v2.11, include several changes to the hardware device modeling layer, a refreshed user interface and some significant changes to Regression Test Manager. To support this investment, Level Four is also expanding its product team with a number of recent hires.

The updates are all aimed at supporting automation best practices and making test case creation and long term maintenance easier. For example, baseline test data and test rules are no longer tied together. Using v2.11, financial institutions can create rules that are completely stand-alone. This change means that fewer rule sets need to be created and if changes are made there is no need to re-execute every test, saving customers' valuable time.

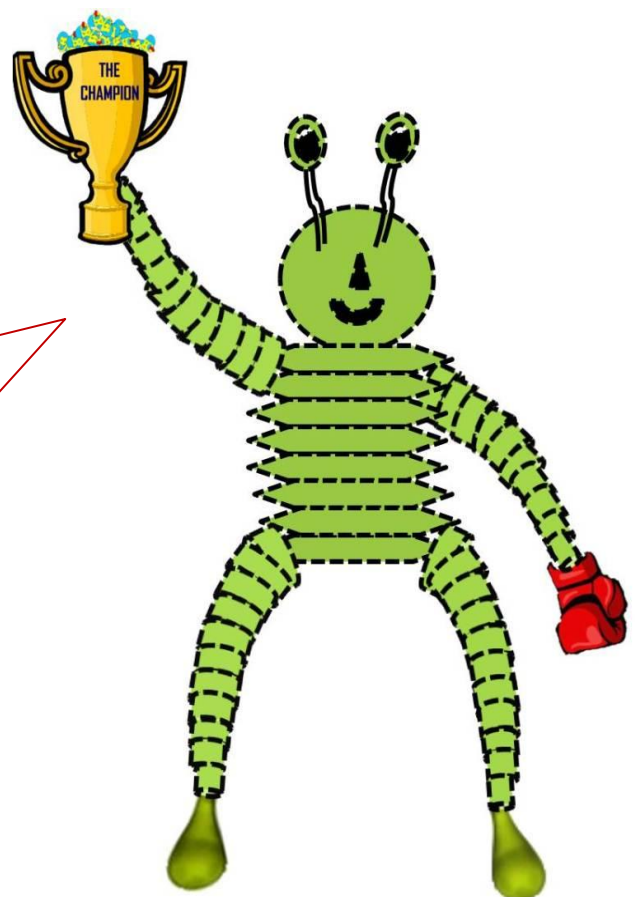
Colin Fraser, Product Development Manager at Level Four, says, "The key points in creating a good test automation system are similar to common software engineering principles. We have increased the ability to share scripts and rules between multiple test cases. This means that when the application inevitably changes, the automated tests can be updated with a minimum of effort. We have also made more provisions for scalability to keep pace with our customers' needs and ensure that test cases are more focused for optimum speed and accuracy."

- Finextra

## Breaking news...!!!

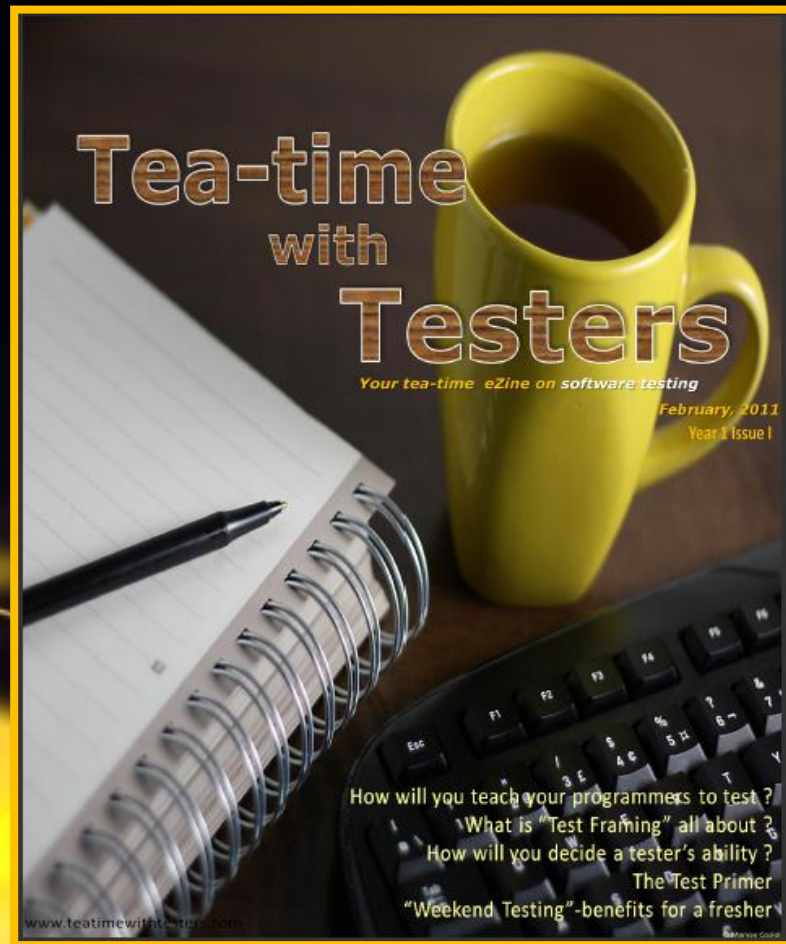
"**Smart Tester of the Month**" &  
"**Blogger Of The Month**" awards  
announced.

Check out our **Announcements** section.



# Feb - 2011

## Our First Issue



300+ Downloads

15000+ Unique Page Views

1000+ Worldwide Readers

[www.teatimewithtesters.com](http://www.teatimewithtesters.com)

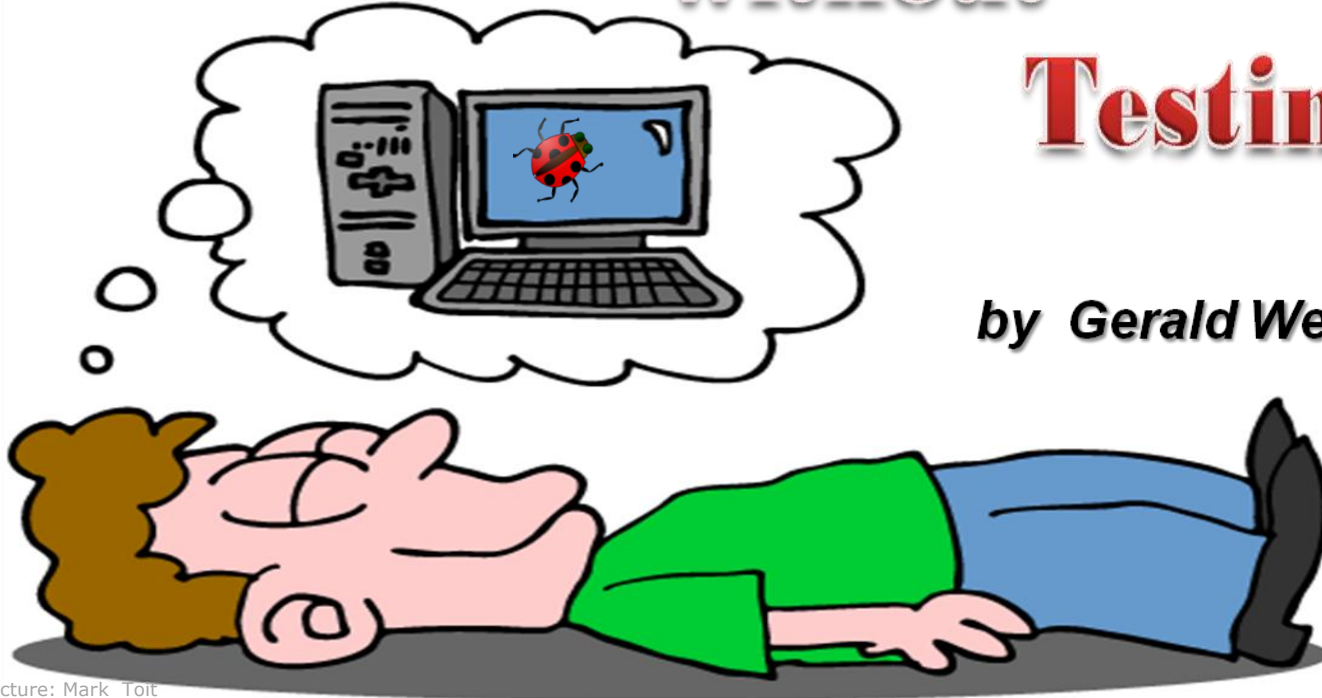


A photograph of a green, conical pendulum bob hanging from a thin wire. The bob is positioned over a surface of light-colored sand. In the sand, there is a large, circular, swirling pattern that resembles a sand mandala or a ripple in water. The text "Speaking Tester's Mind" is overlaid on the image in a large, blue, serif font with a white outline.

# Speaking Tester's Mind

- straight from the author's desk

# Testing without Testing



Picture: Mark Toit

*by Gerald Weinberg*

The job of software testing, we know, is to provide information about the quality of a product. Many people, however, believe that the only way to get such information is to execute the software on computers, or at least review code. But such a belief is extremely limiting. Why? Because there's always lots of other information about product quality just lying around for the taking-if it were only recognized as relevant information.

Because of their psychological distance from the problems, external consultants are often able to see information that escapes the eyes and ears of their clients. Dani (Jerry's wife) tells this story about one of her consulting triumphs in the dog world:

A woman with a Sheltie puppy was at her wits' end because "he always poops on the living room rug." She loved the little guy, so before giving up and taking him to the Humane Society, she came to Dani for a consultation. Dani listened to the woman describe the problem, then asked, "Are there any other behavior problems you've noticed?"

The woman thought about that for a while, then said, "Well, yes, there is one. He has this annoying habit of scratching on the front door and whining."

It's easy to laugh at this woman's inability to see the connection between the two "problems," but that sort of blindness is typical of people who are too close, too emotionally involved, with a situation.





***"Learning to recognize such free information is one of the secrets of successful testing. Using it, you can learn quickly about the quality of an organization's products or the quality of their information obtained from machine testing."***

Here are some "Sheltie stories" from our consulting practices. Test yourself by seeing what information you can derive from them about the quality of a product or the quality of the information that's been obtained about the product through testing:

1. Jerry is asked to help an organization assess its development processes, including testing.

Jerry asks the test manager, "Do you use specs to test against?"

Test manager replies, "Yes, we certainly do."

Jerry: "May I see them?"

Test manager: "Sure, but we don't know where they are."

2. Jerry is called in to help a product development organization's testing group. He learns that they are testing a product that has about 40,000 lines of code.

"The problem," says the test manager, "is with our bug database."

"What's wrong with it," Jerry asks. "Is it buggy?"

"No, it's very reliable, but once it holds more than about 14,000 bug reports, its performance starts to degrade very rapidly. We need you to show us how to improve the performance so we can handle more bug reports."

3. Bunny is asked to help improve the testing of a large product that has 22 components. The client identifies "the worst three components" by the high number of bugs found per line of code. Bunny asks which are the best components and is given the very low bugs per line of code figures for each.

She then examines the configuration management logs and discovers that for each of these three "best" components, more than 70 per cent of their code has not yet successfully been checked in for a build.

4. Trudy is invited to help a development manager evaluate his testing department's work. She starts by looking at the reports he receives on the number and severity of bugs found each week.

The reports are signed by the test manager, but Trudy notices that the severity counts have been whited out and written over.

"Those are corrections by the product development manager," the development manager explains.

"Sometimes the testers don't assign the proper severity, so the development manager corrects them."

Using her fingernail, Trudy scratches off the whiteout. Under each highest severity count is a higher printed number. Under each lowest severity count is a lower printed number.

**5.** Jerry is watching a tester running tests on one component of a software product. As the tester is navigating to the target component, Jerry notices an error in one of the menus. The tester curses and navigates around the error by using a direct branch.

Jerry compliments the tester on his ingenuity and resourcefulness, then asks how the error will be documented. "Oh, I don't have to document it," the tester says. "It's not in my component."

**6.** Fanny watched one tester spend the better part of several hours testing the scroll bars on a web-based enterprise system. The scroll bars were, of course, part of web browser, not the system being tested.

**7.** Jerry asked a development manager if the developers on her project unit tested their code.

"Absolutely," she said. "We unit test almost all the code."

"Almost?" Jerry asked. "Which code don't you unit test?"

"Oh, some of the code is late, so of course we don't have time to unit test that or we'd have to delay the start of systems test."

**8.** One of Christine's clients conducted an all-day Saturday BugFest, where developers were rewarded with cash for finding bugs in their latest release candidate. They found 282 bugs, which convinced them they were "close to shipping."

They were so happy with the result that they did it again. This time, they found 343 new bugs - which convinced them they were "on the verge" of shipping.

**9.** A general manager was on the carpet because a recently shipped product was proving terribly buggy in the hands of customers. Jerry asked him why he allowed the product to ship, and he said, "Because our tests proved that it was correct."

**10.** Another manager claimed to Noreen that he knew that their product was ready to ship because "we've run 600,000 test cases and nothing crashed the system."

**11.** When Jerry asked about performance testing, one of his clients said, "We've already done that."





"Really?" said Jerry. "What exactly have you done?"

"We ran the system with one user, and the response time was about ten milliseconds. Then we ran it with two users and the response time was twenty milliseconds. With three users, it was thirty milliseconds."

"Interesting, Jerry responded. "But the system is supposed to support at least a hundred simultaneous users. So what response time did you get when it was fully loaded?"

"Oh, that test would have been too hard to set up, and anyway, it's not necessary. Obviously, the response time would be one second - ten milliseconds per user times one-hundred users."

**12.** Jerry's client calls an emergency meeting to find out "why testing is holding up our product shipment." In the meeting, the testers present 15 failed tests that show that the product did not even build correctly, so it couldn't be tested. They discuss each of the 15 problems with the developers, after which the development lead writes and email summary of the meeting reporting that there are only two "significant" problems.

The email goes to the development manager, the marketing manager, and all the developers who attended the meeting. None of the testers present are included in the cc-list, so none of them even know that the email was sent.

**13.** Johnson watches a tester uncover five bugs in a component, but instead of logging the bugs in the bug database, the tester goes directly to the developer of the component and reports them orally. Johnson asks why the tester didn't record the bugs, and he replies, "If I do that, she (the developer) screams at me because it makes her look bad."

**14.** Tim reviews a client's test plan and notices that there is no plan to test one of the components. When he asks the test manager (who is new to the company) why it's missing, he's told, "We don't need to worry about that. The development manager assures me that this developer is very careful and conscientious."

So, were you able to extract meta-information from these ten examples? What did each of them tell you (or hint to you) about the quality of the information from testing - the accuracy, relevance, clarity, and comprehensiveness? Remember, though, that these are merely hints. Perhaps the Sheltie pooped on the rug because he had some medical problem that would need a veterinarian's help.

Perhaps there's a non-obvious explanation behind each of these Sheltie Situations, too, so you always have to follow-up on the hints they provide to validate your intuition or find another interpretation. And, even if your intuition is right on target, you probably won't have an easy time convincing others that you're correct, so you will have to gather other evidence, or other allies, to influence the people who can influence the situation.

When Dani asked the Sheltie's owner why she thought the pup was whining at the front door, the woman said, "I think I've spoiled him. He just wants to go out and play all the time, but I have too much housework to do - especially since I spend so much time cleaning up the mess on the rug."

When a problem persists in spite of how obvious the solution is to you, you aren't going to be able to convince others to solve the problem until you find out how they are rationalizing away the evidence that's so apparent to you. So we need to know how people immunize themselves against information, and what we can do about it.

## Biography



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

Gerald is PhD in Communication Sciences from the University of Michigan.

He started working in the computing business at IBM in 1956 at the Federal Systems Division Washington, where he participated as Manager of Operating Systems Development in the Project Mercury, which aimed to put a human in orbit around the Earth.

In 1960 he published one of his first papers. Since 1969 he is consultant and Principal at Weinberg & Weinberg. Here he conducts workshops such as the AYE Conference, The Problem Solving Leadership workshop since 1974, and workshops about the Fieldstone Method. An author at Dorset House Publishing since 1970, consultant at Microsoft since 1988, and moderator at the Shape Forum since 1993.

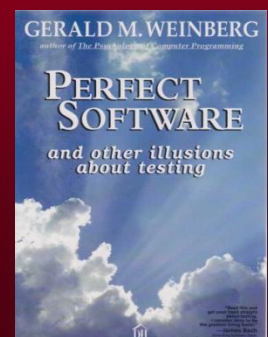
Gerald has been a visiting professor at the University of Nebraska-Lincoln, SUNY Binghamton, and Columbia University. He has been a member of the Society for General Systems Research since the late 1950s. He is also a Founding Member of the IEEE Transactions on Software Engineering, member of the Southwest Writers and the Oregon Writers Network, and a Keynote Speaker on many software development conferences.

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](#) .

To read one of his best-selling books "**Perfect Software and Other Illusions about Testing**" please click [here](#) or read it online [here](#).

Gerald can be reached at [hardpretzel@earthlink.net](mailto:hardpretzel@earthlink.net) or on twitter @JerryWeinberg





A green frog with large, white eyes and a small orange nose is perched on the edge of a brick well. The well is made of brown bricks with a pattern of lighter brown spots. The background is a dark blue gradient.

# Why good testers should come out of the well...?

*by Pradeep Soundararajan*

**O**ne problem that I constantly spot whenever I meet a good tester in India is, they don't blog and publicly write or speak. That's why some people continue to think about me as a good tester.

The recognition that an organization gives could be fine as long as you stay in the organization or as long as the organization is doing fine. That's why many people stick on to an organization for quite sometimes because they start hating fresh water. They accept to be in a well. With growing infrastructure needs and changing economy, some wells have to be wiped out and all frogs in it have to hop to a different destination.

I was once surrounded by bad frogs who made me feel that I was a great tester. I would never want to be there again because that hampers my learning although it pleases my ego as long as I am with them. A couple of months back I interviewed a tester who had been awarded as the best among 1100 testers of his organization. He was pathetic and I think the right one in the organization didn't get the award.

A tester from Mumbai who claimed to be superior to me in knowledge and skills wrote to me and said, "You are misguiding the community by giving wrong ideas" and my reply to him is this, "Well, if you are so concerned about the community then you should write a blog and say to the world that Pradeep Soundararajan is giving wrong ideas and the reasons of why his ideas are wrong" for which he never got back to me with his blog link.

***There are several testers whom I helped to start a blog and only some of them are doing fine with it. Some people started a blog and sent me a link with a note that - you inspired me. If I revisit their blogs, most of them ended up not continuing it because they realized it is hard to keep blogging. The other dimension is - it is easy to blog if you are just doing a cut copy paste plagiarize, not owe credits to original authors and expose yourselves as a fool.***

2 months back, I interviewed Harish, who had lost his job from a reputed organization which decided to shut down its operations in Bangalore as they faced the worst part of recession at their US office.

Harish is the kind of tester whom I'd want to work with for the way he challenged my arguments, sharp eyes that observes little things going around the screen and has good reporting skills, good communication skills, but then, my client had to postpone their recruitment plans. I couldn't get an opportunity to work with him. If you are looking for one, I'd suggest you talk to Harish. I wish I could have linked to his blog to get you curious about him and that's what is missing.



I asked him

PS: You don't blog?

Harish: Why should I?

PS: For the world to know about a good tester.

Harish: Why should the world know about me?

PS: Consider asking yourselves as to why shouldn't the world know about you?

Harish: Let me think about it. After a month, Harish calls up, "Hey Pradeep, I haven't found a job yet. I realize this wouldn't have happened if the world knew about my testing skills. The interviews test something different than my skills". So here is an article for all those Harish of the world to wake up and start blogging. A blog of your own serves a core purpose that surrounds all of us - ***to learn - things, ways, people, testing, ideas, challenges, and more...***



## Myths that surround wannabe-tester-bloggers

### If I should blog, I should have good writing skills:

Ha! You should read [Pradeep's first post](#) and then you would realize that he was more pathetic in writing than what he is today (or what you might have been a couple of years back). However, as you peruse through the blog you would realize that I have improved a thousand leaps. It comes from practice and a blog helps you to practice writing. Two things never happen to people with this myth - better writing skills and blog.

### If I should blog, I should be an expert:

I must admit that I thought of myself as the world's best tester till I met James and then more people like you. My blog has helped me meet thousands of people who helped me understand that I am not the world's best tester. That's important to learn because it gives me learning opportunity to try to get as close to what I think I was. You don't need to be an expert to blog but people commenting on your posts can help you to be an expert of the field. They might surprise you with a question that you think over for the next 2 years to find answers for it and in search of an answer to that question you discover a whole new world of testing.

### If I should blog, I should have thousands of readers and comments:

I am my first blog reader. I primarily write to practice writing and thinking. I have been writing this post over 2 days and I test my writing. It helps me writing some good documents at work that influences decisions. If my writing can be of help to others then I am glad. I do not write to get thousands of readers or comments. No matter the world stops reading my blog, I would continue to write for one regular and serious reader - that's me.



### If I should blog, I should write in a way that fetches appreciation:

Saurav Ganguly, a cricket player from India was axed out of the team for poor performance a couple of years ago. The internet in India was full of jokes about his poor performance but then he made a great comeback to the worldcup squad. He was interviewed in NDTV for his comeback and a journalist asked him, "How did you make this great comeback when most parts of India, including Kolkata, your hometown was against your

performance?" to which he replied, "I didn't spend time bothering things that are not under my control ( people making fun and jokes about his performance) and focused on spending more time for things that are under my control

( practice, improving performance, consistent results in league matches) and I think that's what helped me". I took this as a great lesson to myself and focused on doing things under my control and not bothering things that are not under my control.

### **If I should blog, I should have more time than what I have:**

Actually, we spend time on lot of useless things every day. If you cut that out of one day in a week that provides you time to blog. I usually laugh at people who say they have no time to blog. I woke up a little early today to complete this post. I think I wouldn't die if I get up early.

As James Bach said on Twitter:

***"I don't teach my son the value of discipline and hard work, because that *\*can't\** be taught-- only learned."***

### **If I should blog, I should be a good tester:**

A real good tester would want to get tested to see if he is really good and would be glad to know he is not good since that helps in improving him. A blog is probably one of the ways in which you get to know about the holes in your education as a tester. Once you know that, start plugging them. You could aim to be a good tester and start blogging than wait for you to become a good tester and then start blogging.

There are number of other ways you can demonstrate the world about your testing skills and I think you should do that. Ah! No, not by saying you are proud to be ISTQB certified, that would drive away people.



**PhilK** recently interviewed many tester bloggers and I was one of them. Read my interview [here](#) and do not forget to read other interviews as well.

If you are good, the world should know about you. If you are hesitant to let the world know about you - you aren't good enough, maybe. Good tester doesn't mean you offer advice to the world, it means you present your work and be open to learning from others if they happen to argue. You may be a good tester and choose not to write a blog, I still respect that but I think you can get better by writing one.

So, time for you to get back to your work, ignoring all things in this article and continue to say, "If I should blog". If you are already doing it get more people to be like you or do more with that.



## Biography

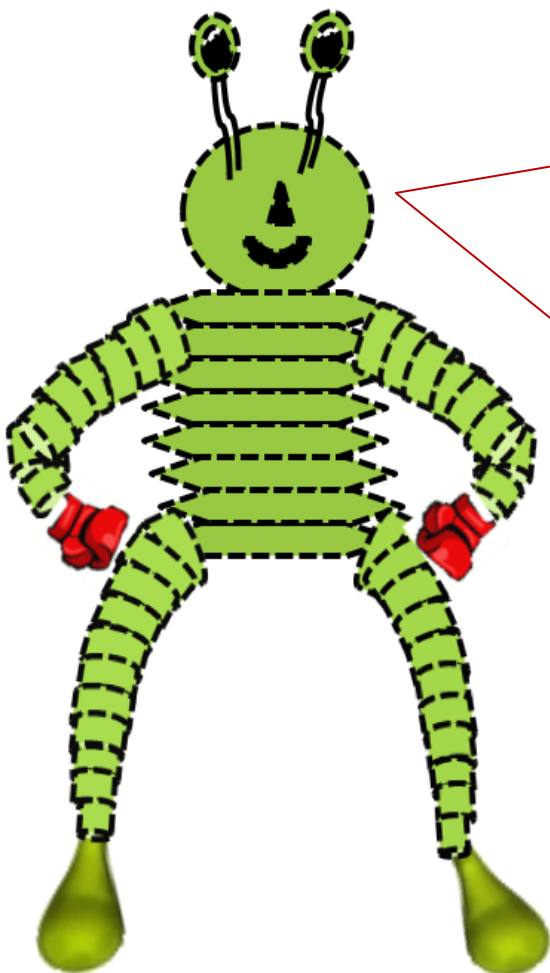


**Pradeep Soundararajan** is Co-Founder, Director of Moolya Software Testing Private Limited.

He is a well known tester and blogger in the Testing World. He is a **Test Consultant, Coach, Speaker and Mentor** to many testers across the world.

Pradeep writes his experiences in Testing at his own blog [TesterTested](#) .

He can be reached at [pradeep.srajan@gmail.com](mailto:pradeep.srajan@gmail.com) or at twitter [@testertested](#) .



**Cool. Pradeep has said it all...!!!**

**Guys, do you still have any issues in starting up your own blog now???**

**Well, come on. It's not that tough you see.**

**So who all are writing for this month's challenge? I will see your entry fellows.**

**Mail me your articles/blog entries and claim your right on '**Blogger of the Month**' award.**

**Ciao...!**

**Don't know about **Bug-Boss Challenge**? Click [here](#)**

*Are you just a Tester ...  
or a writer, painter & musician too?*

# managing multiple passions

*by Anuj Magazine*



**Harsha Bhogle** in his book "Out of the Box - Watching the Game We Love" has a mention of **Peter Reobuck** who told the twin loves of his life: cricket and the English language. They work well together; certainly in our game which lends itself to fine writing. The above lines have a unique reflection about people having multiple passions and interests. Peoples' talents are usually not limited to being good at just one thing for life (we usually refer to as Passion) and are helped immensely by parallel interests. In case of Peter Reobuck, it's Cricket and English language.

**Jack Russell** came into limelight when was a part of English cricket team in 1980s and 1990s. Though not being formally trained at art, he had a special interest in art and took up as a full time artist after his Cricket career was over. Not only this, he also is a Football goalkeeping coach.

In **Michael Phelps** biography (Michael Phelps-The Untold Story of a Champion) by Bob Schaller, there is a mention of fellow swimmer **Natalie Coughlin**. He says "A graduate of Cal-Berkeley, Natalie Coughlin has interests outside of the pool that means as much to her as her life in water. She goes out of her way to keep balance in her life, and while swimmers everywhere do doubles and train hours every single day, Coughlin is just as prone to balance a single daily workout with Pilates, walking her dog or Spinning."

***"Looking at the Blogosphere, there are so many distinguished testing practitioners who not only are the experts at their craft but also find time to do write frequently in blogs, magazines, speak in conferences and other relevant forums "***

There exist people who seemingly achieve more in a day thereby giving an impression to the external world that God probably was more kind to these people and gave them more than 24 hours in a day. It is often intriguing to me to wonder why some people manage different passions so well while some people make an attempt at the same but with little success.

My mind had been working overtime to demystify this fact and following in this article. I would like to share some thoughts around it. The text that follows is mostly in the form of answers to some relevant questions involving handling of multiple passions.

### **Am I "Only Interested" or Am I "Fully Committed"?**

Ken Blanchard in his book "The Heart of a Leader" mentions-I learned from author and consultant Art Turock that we need to make a distinction between being interested and being committed. When you are "interested" in doing something, you only do it when it's convenient, but when you are "committed", you follow through no matter what, no excuse."

I think this is a very relevant observation. People who manage multiple passions are not only interested and deeply in love with their areas of interests but are also hugely committed towards the same. They operate in a mode as if it's their responsibility to contribute towards their passion every day, every hour. Many people get inspired and interested in pursuing additional interests but fail to pursue it for a longer period of time because the convenience factor creeps in with the lack of commitment. If one is committed to the cause, it's not impossible to cross any obstacle. If you like to do something but haven't been able to pursue, just ask- Do I really want it badly? Am I really only interested or committed also? The sooner one sorts out the answers to these questions, it lends the desired clarity to live with one's passions.



### **Am I able to "prioritize" effectively?**

I was having a conversation with [J.D.Meier](#) about managing multiple priorities and why individuals even while having interests in multiple activities aren't able to contribute enough or balance properly.

He said-Here's what I do: Mapping out what's Important: Identify the most important results in each area or hot spot in your life (just the top 3 result for each hot spot or interest)



### Producing Results:

- On Mondays, I identify 3 results for the week
- Each day, I identify 3 results I want to accomplish (this drives my day)
- On Fridays, I reflect on my results. I identify 3 things going well, and 3 things to improve.
- Each month, I pick a theme for focus. This is how I can balance across my interests. It allows me to focus less on one thing, while I focus more on another.

The question may arise, whether an individual should talk about more than 3 priorities at a given time? J.D. Meier answered it: Why not finish the first 3, and then bite off more? The value of 3 is that you can remember it without writing it down. Test yourself. Today, I had 3 priorities - complete my draft vision, confirm the budget, and sync w/my partner group. Did I have lots more things to do? ... Sure, but those 3 were my best bets for the day. If I got through those 3, I could always bite off more. What I didn't want to do was have a long list of things I couldn't remember.

The rule of 3 was actually found to be the most effective number in the military for people to remember outcomes without writing things down. I didn't know this at the time. I'm just happy that the military came to the same conclusion. I lucked into it :) Isn't this Simplification at its best? Clear mind does achieve more than a cluttered mind.

### Am I able to "create" enough time?



Of course, the common thought people who want to do more but aren't able to do so is that they don't have enough time. From my observations, the idea is to create time for self, for what you want to do. There are infinite ways one can look at the day and build up time for something you really want to do. One of the ways, Subroto Bagchi mentions in his book "The Professional" when he introduces the term- "White Space"- I do not know how the term White Space originated. In telecommunication, it denotes frequency allocated to a channel but not used. Typically, broadcasters are provided additional frequency that is not meant to be populated so that the adjacent broadcast stations do not overlap. In print, the white space denotes emptiness so that we can read between the characters that

form a word and a group of words that form a sentence. In our professional lives, white space is a train or a bus ride to work; it is the time waiting outside the client's office, the time spent on long flights. We have all been given huge white spaces and we simply let their power go waste.

Look back at the day you spent, you will notice the white spaces and how you utilized them. If you wanted to read a book, could you have utilized the white space better? If you wanted to write something, could white space have been utilized in a better manner? I am not suggesting you to use all the available White spaces towards your interests, probably they are not ideal. But what is more

apt is that in order to create time for self, White space offers the necessary time space for you. By just having a look at how you utilize the time in the day, you would be able to figure out how to manufacture time. Believe me, it's possible!

### Am I able to "compartmentalize" life?

"Our main business is not to see what lies dimly at a distance, but to do what lies clearly at hands" - Thomas Carlyle Dale Carnegie too was a huge proponent of Living in a Day tight compartments. Over a period of time, I have begun to appreciate the essence of living in Day tight compartments. Much of our mental and emotional energies drain out worrying about what will happen tomorrow or also by replaying the mistakes that we did yesterday in our minds. Living in a day-tight compartment is a metaphor that symbolizes that we shut out thoughts that carry over from yesterday or thoughts that represents tomorrow's anxieties and live life as it happens today. The idea is not to let residue thoughts of yesterday and toxic worries of tomorrow bother you today.

I have found the application of this concept suitable even while managing multiple passions. I try and live a slightly modified version of this principle- "Live in an hour tight compartments" when you are dealing with multiple interests. It's usual that disappointment resulting in failure or a mistake in one task during the start of the day often occupies your mind throughout the day and affects everything else you do throughout the day. Living the day in a hour tight compartment helps one shut out what happened in the earlier on and helps maintain the required focus in other interests.

## Biography



**Anuj Magazine** is a Software Testing practitioner currently working at Citrix R&D India Pvt. Ltd. He likes exploring new avenues and their intersection with Software testing profession such as Handwriting Analysis, Assembly line approach, Six thinking hats etc. He is passionate about Software testing and has been a regular speaker at conferences, such as QAI, STEPIn forum, SoftTec etc. and has spoken on several software testing topics. He has written articles in [www.stickyminds.com](http://www.stickyminds.com), TheSmartTechie Magazine, and writes frequently in his blog Creative Tester .

Anuj also holds a professional qualification in Handwriting Analysis. He can be reached at [amagazine@gmail.com](mailto:amagazine@gmail.com) & at twitter [@anujmagazine](https://twitter.com/anujmagazine)

A photograph of a classroom scene. Several students are seen from behind, sitting at desks and raising their hands. The background is a chalkboard with some faint writing. The image is framed by a thick black border.

# In the school of Testing

*for your better learning & sharing experience*





# turning the tide of Bad Testing (Part I)

by Martin Jansson

***“Social psychologists and police officers tend to agree that if a window in a building is broken and is left unrepaired, all the rest of the windows will soon be broken. This is as true in nice neighborhoods as in run-down ones. Window-breaking does not necessarily occur on a large scale because some areas are inhabited by determined window-breakers whereas others are populated by window-lovers; rather, one unrepaired broken window is a signal that no one cares, and so breaking more windows costs nothing. (It has always been fun.)”***

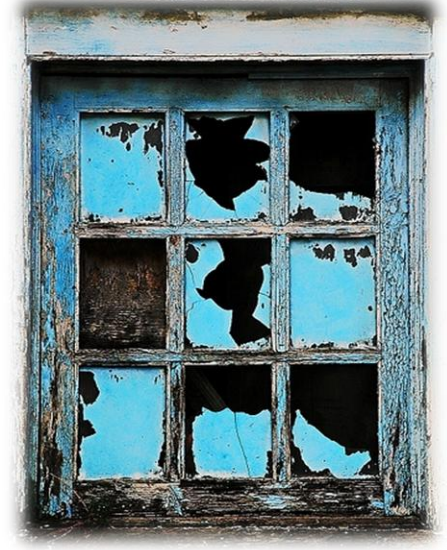
**A** quote from an article published 1982 in [Atlantic Monthly](#) and was written by James Q. Wilson and George L. Kelling. This metaphor can be applied to many situations in our daily lives. It is often used in sociology and psychology.

At [Artima Developer](#) , Andy Hunt and Dave Thomas, authors of *The Pragmatic Programmer: From Journeyman to Master* (Addison-Wesley, 1999), discuss and elaborate around an important part of being a Pragmatic Developer namely fixing Broken Windows.

The book mentions many ways of keeping clear of the broken window in order to avoid an increasing technical debt.

***“One broken window – a badly designed piece of code, a poor management decision, that the team must live with for the duration of the project – is all it takes to start the decline. If you find yourself working on a project with quite a few broken windows, it’s all too easy to slip into the mindset of ‘All the rest of this code is crap, I’ll just follow suite’”***

I’ve seen developers focus on getting the amount of warnings down. I’ve seen project managers addressing the flow of bugs, thus measuring the fix rate, find rate, resolve rate and close rate to keep the project from becoming unhandlable. Another project manager was stating that developers must fix “shitty bugs” or the smaller bugs, before the build reaches the official testers. We see different treatments of this in development.



### **How do we see Broken Window Theory affect testing?**

- When you have stopped caring about how you test.
- When you have stopped caring about how you report bugs and status.
- When you have stopped caring about how you cooperate with others.
- When you have lost focus on what adds value.
- When you do not cooperate with developers and have stopped talking to them.
- When you complain about the requirements and have stopped talking to the business analysts.
- When you avoid testing areas because you know that bugs won’t get fixed there anyway.
- When you avoid reporting bugs because it doesn’t matter.
- When you report status as you always have been, without any real content.

... and so on...

All these create Broken Windows and, as I see it, result and are summarized in a Testing Debt, which was inspired by Ward Cunningham’s definition of [Technical Debt](#). Jonathan Kohl has made a definition of [Testing Debt](#) and Johanna Rothman [another](#).

### **How do you identify things that increase this Testing Debt?**

Before knowing about, ‘Decreasing Testing Debt’ it is equally important to learn about identifying the things that increase it.

Here are some ways I would suggest to identify the things that increase the Testing Debt:

In your current assignment as a tester, test lead or test manager think about how the situation is for the test group right now.



### **At the stand-up morning meeting with the test group, stop and consider...**

- What is that troubles the group?
- What obstacles do they see?
- What is their main concern?
- What is stopping them from performing their work?

### **At lunch with your co-workers, stop and listen...**

- What do they complain about?
- What is making them frustrated?
- What is the work related topic that comes up every so often?

### **At the meeting with business analysts, stop and consider...**

- Do you think the test group has better knowledge of the customer than those writing requirements?
- Do the business analysts ever ask what the test-group thinks about new features?
- Do you feel that the cooperation and collaboration is good with the business analysts?

### **At the meeting with project management, stop and consider...**

- What issues are brought up at almost every meeting but are usually ignored?
- What risks get lower or higher attention than others by the project manager?
- How is the relationship between the project members and the project manager?
- What risk areas are always brought up at every project and always come true?
- Have you communicated your belief in the project plan?
- Do you feel that cooperation and collaboration is good with the project members and the project manager?

### **When discussing with developers, stop and consider...**

- Do you provide what they are asking for?
- Is there a conflict growing here? Are you nurturing the conflict?
- Do you feel that the cooperation and collaboration is good with the developers?



### **At the line meeting with the test manager and the rest of the test group stop and consider...**

- Is the focus different on issues than that the project[s] think are important?
- Is the focus on getting better as a test group or is it clouded by other issues, what are these in that case?
- How many in the test group have been degraded down to a tester and are now stuck there, as they see it?
- Do the internal conflicts in the test group outgrow the external conflicts?
- Do you feel that the cooperation and collaboration is good within the line?

### **When you are testing, stop and consider...**

- Are there any bugs you ignore and choose not to report?
- Are there any areas that you know bugs won't get fixed in that you choose to avoid?
- Are there any areas that you have to little knowledge about that you choose to avoid?

### **Before you are about to submit your bug report, stop and consider...**

- Have you considered how many stake holders will look at this and how much time they will spend reviewing it? Do you ignore that fact?
- Do you think the developer will be able to fix the bug with all information that you have provided?
- Do you spend minimal amount of time possible on writing the report?
- Do you care for and take pride in the bug report?
- Do your bug reports usually get fixed or are they returned because of lack of information?

### **Before you send your status report about testing, stop and consider...**

- Have you provided truthful information, as you see it, that you think will be valuable to the reader?
- Have you emphasized any area that you personally just want to get more focus?
- Have you emphasized any risk that is not that risky?
- Do you think stakeholders would be able to make good decisions based on the information you have provided?

Now, what is the trend you see? What picture have you painted? The next hard thing is what you intend to do about it. Remember, it is often good to start with yourself.

Let's talk on "**Decreasing the Testing Debt**" in my next article in **Tea-time with Testers - April Issue**.

## Biography



**Martin Jansson**, Test Manager at Qamcom Research & Technology, 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](http://www.thetesteye.com) which has grown into one of the greatest Swedish blogs on software testing. In 2010 he and a colleague won the competition for apprenticeship in EuroSTAR TestLab and they will together in 2011 manage it. Martin is always on the lookout for great, passionate people to work with.

You can reach him on Twitter [@martin\\_jansson](https://twitter.com/martin_jansson) or on [martin.jansson@qamcom.se](mailto:martin.jansson@qamcom.se)

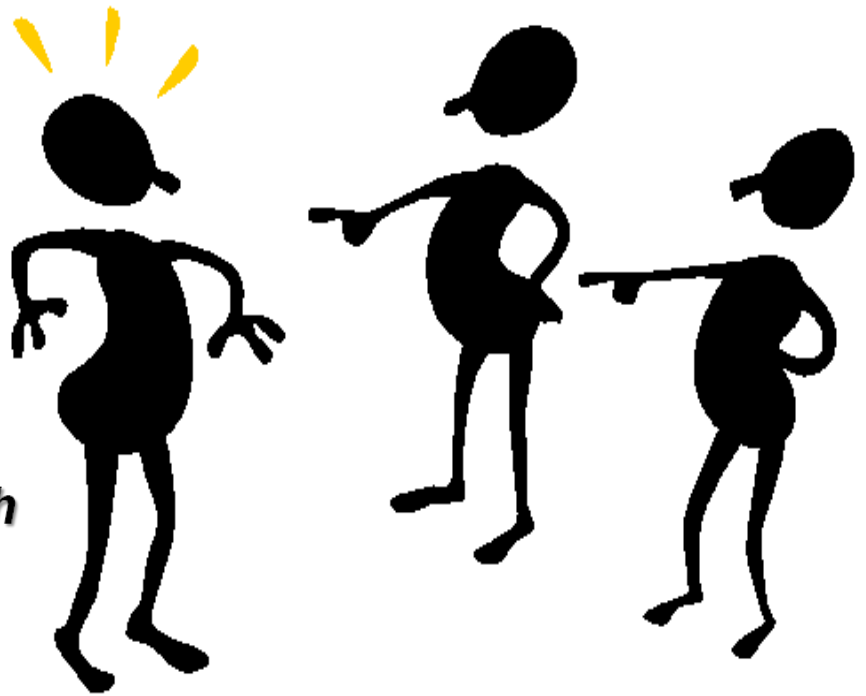


Confused about writing down good Test Cases..!

Chill...! Read Joel's article 'If you are going to write down a test case'. A must read for fresh testers. Click [here](#) to jump.

# when QA gets blamed

by James Bach



**Rohit** approached me with a classic problem: what do you do if you are blamed for not finding a bug, given that you were faced with a bad specification, spent a lot of time to help fix it, and then didn't have enough time to get the work done? There are lots of answers, here, but I thought I'd start with a simple answer to see how he'd react.

**James:** This is not a testing problem. This is a straight forward human problem. All you have to do is say "This is not our problem." What happens when you say that?

**Rohit:** Meetings, escalations. But is there any way out for this? Why is it that analysts miss so often and QA has to do everything? How to cope up with the frequently changing requirements and still deliver better in time?

**James:** One of the problems is that you are accepting the blame. Don't accept it. Just laugh. Ask them if they also want to blame you if the rains don't come... or if it rains too much! However, they may not be blaming you at all. Perhaps, it just feels that way. Perhaps, they just want you to save them. They are hoping that you can save them.

**Rohit:** They asked why QA raised this defect so late?

**James:** And what's your answer?

**Rohit:** In so little time, with all analysis work done by QA, and also the sincere testing of it, it's human to miss something. We did not test that particular scenario.

**James:** Did they say you made the product late?

**Rohit:** No, they said it's intentional that we have raised it late, so as to harass development.

**James:** Ah, they are saying that you knew about the problem and hid it from them? Who is making this claim?

**Rohit:** The developer and developer manager.



**James:** They have no evidence that you knew about it earlier, right?

**Rohit:** Right. Also, they are asking if not known what QA was doing until late?

*Here Rohit has given me a clue that the work he did was not done with the full knowledge and support of his clients. This actually might be his problem, after all. I have to probe deeper.*

**James:** That's a good question. Why did you not find it? You should be able to answer questions like that easily.

**Rohit:** We found it late because we were overloaded with analysis work along with testing.

**James:** What if you hadn't done that analysis work?

*I ask this question because I want to know if the analysis was part of his job or not.*

**Rohit:** We could have very well found it.

**James:** Then why did you do the analysis work? If the analysis work gets in the way of your testing, you could simply not do that work.

**Rohit:** Okay, the analysis part: without QA's analysis it was impossible to deliver correct application.

**James:** But what does that have to do with testing? Were you asked to do analysis? Was that your job?

**Rohit:** If requirements are not correct, how can we test it? It was the analyst's job. He was sleeping. The document he had written was mere reflection of that. That was something we could not accept, having better knowledge of the application and the requirements.

**James:** Were you asked to do that work?

**Rohit:** No.

*Ah, that's why they can blame him for not finding any particular thing.*

**James:** Who do you work for? The dev manager ?

**Rohit:** Client Manager. The dev manager reports to him.

**James:** Does the client manager expect that you will do this analysis work?

**Rohit:** He appreciates if QA adds to quality.

**James:** That doesn't answer my question. Does the client manager expect you to perform analysis? I want to know if you are off on your own, doing things you weren't asked to do. Because if you are, then of course you will get in trouble.

**Rohit:** Yes, officially we are not getting billed for analysis. We are doing it pro-actively. But being a tester my question is, shall we compromise?

**James:** When you go outside of the contract you have with your manager, then you are taking a risk. It's simple: If you want to keep your job, do what you are hired to do. If you can't bring yourself to do that, then expect to be blamed for not doing it.

**Rohit:** Well, when we say QA should be involved as early as possible, what is wrong to analyze things?

**James:** What's wrong is that you don't have a charter to do what you want to do. So, get a charter. Get your manager to tell you to do the analysis. Get him to agree with the way you are using your time. Then when you are "blamed" for not doing everything, you can point to your manager and say "take it up with him." I went through something like this last year. I worked for a client that didn't support me in doing what I thought I should do. I stopped doing that stuff.

You can't get too far ahead of your client.

**Rohit:** Why is it that clients listen to dev folks most of the time?

**James:** Oh, that's easy. You haven't built credibility. The developers have more credibility. You build credibility by dealing with difficult problems and showing that you solve them well. But you have to do that without running wild (in the client's mind).

You have to maintain contact with your client.

*If I were you, I would spot the need for the analysis work, and I would go to my manager and tell him what I thought I should do and would warn him what would happen if I spent time on that. And I would warn him what would happen if I didn't spend time on that. Then I would make my recommendation and ask what he wanted me to do.*

*I do think that testers can and should take initiative and help wherever they can. But if we do that in conflict with the expectations that our clients have for us, then we have little defense against the charge that we wasted time instead of doing our job.*

## Biography



**James Bach** is a software tester, author, trainer and consultant. He wrote numerous articles for IEEE Computer. In Software Testing, he is a proponent of the **Context-Driven school** and advocate of **Exploratory testing**. He is credited with developing Session-based testing. According to Google Scholar, several of his articles have been cited dozens of times including his work on heuristics for testing and on the Capability Maturity Model. He is a member of the Board of Directors of **the Association for Software Testing**.

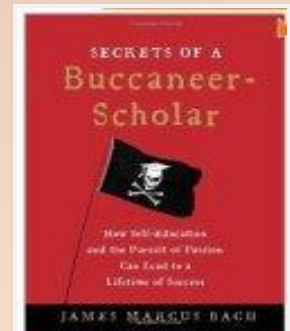
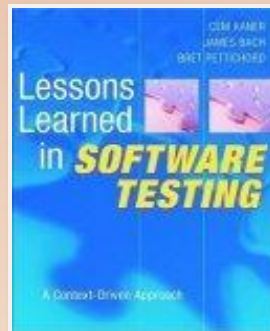
Since 1999, he works as independent consultant out of Eastsound, Washington. On this basis, he was one of the expert witnesses in the Microsoft antitrust case: he determined that Microsoft could indeed unbundle Internet Explorer from the Windows operating system.

His book "**Lessons Learned in Software Testing**" has been cited over 130 times according to Google Scholar.

James has also written "**Secrets of a Buccaneer-Scholar**".

To know more about James, please click [here](#) or [here](#).

James can be reached at [james@satisfice.com](mailto:james@satisfice.com) or on twitter [@jamesmarcusbach](https://twitter.com/jamesmarcusbach).



# testing intelligence

- *its all about becoming an intelligent tester*



an exclusive series by **Joel Montvelisky**

## If you're going to write-down a test case...

**W**riting down a good and smart test case is a very important thing a tester should be skilled in.

**if you are going to write-down just one (more) test case, what would that test be?**

Will you write a very descriptive (and deep down) test for the most risky component/screen in your application? Or would you write an end-to-end sanity/smoke test that goes over 80% of your system?

This is a question to ask whenever you sit to write scripted testing scenarios.



© Mystic Arts, LLC



## Since you don't have all the time in the world, think about ROI

My guess is that about 95% of the testers reading this blog belong to organizations where time is not a resource to spare. This means you are usually running behind schedule, trying to figure out what tests can be left out and still be able to provide good visibility into the application you are testing.

Don't feel bad about it, these are the rules of the game. Still, this means that regardless how good your intentions are and how much time you set aside at the beginning of your project, you will NEVER have time to write-down all your deep-down functional test cases.

If this is your reality too, better to be smart about it and write down the test cases that will bring you the best Return on Investment (ROI). Some of the working guidelines I have in place to assure I do this (as much as possible) are the following:

### 1. Work based on Risks and Priorities

When you have time to write tests, think about your biggest priorities and start with them. It is important not to look only at priorities based on your project timelines, but also based on the risk factors of your application. It may be more important now to write a test for a high-risk feature that will be delivered in one month than for the mid-risk feature you're getting next week; the reason is that later you may not have a chance to write the test at all. There is no magic formula or excel sheet to define what test to write next, but make sure to keep all factors in mind.

### 2. Use the 80/20 rule

20% of your tests can cover 80% of your application, this is the principle behind sanity and smoke tests, you can write a relatively short scenario that will uncover a significant part of your bugs.

So here is the answer to question at the beginning of the post: If I can write a deep-down test for the riskier component of my app or a sanity test to go over 80% of the complete system, I will go for the later. I'll always write the end-to-end sanity suite and only then start covering specific features. Not only that, when covering a specific feature I will also start by working on the short tests that cover it end-to-end and make sure I have done so for most of my system before moving to go deeper in a single place.

### 3. Write tests you can give to other people to help you test

A factor in choosing what test to write is to work on the stuff that will help you "outsource" your tasks. For example, write tests you can give to developers to run when they are already done with their features and you are neck-deep in testing high-risk tasks you cannot give to anyone else.

### 4. Work based on iterations

You won't finish all your writing in one sit, write your tests so that they are "good enough" for now and know you'll revisit them in the future if you need to improve them. AGILE is not something to be implemented only in development 😊

## 5. Think long range and not short range

If you decide to write down a test, make it so that it will still be valid in future sprints/builds/releases; don't write tests that are too specific and will only be good for the current stage of your project.

### Some practical rules of thumb

#### - Write down the names of your tests before you start writing them.

This will help you to get the big picture and prioritise based on the stuff you need/want to do. Don't worry too much about having a perfect list from the beginning, it will be perfectly OK later on to take one test and break down into 3 or 4, or to take 2 tests and merged them into one. What you want is a list of topics and their risk-based priority.

#### - Divide and catalog tests into Business Scenarios and Functional Tests.

Both of them are important but each looks at the system under a different light. The idea of differentiating is to know what test to write and when to write it; and also to help you organise them as part of your Test Library and to help you (and others in your team) in the future to choose what tests can be run based on the needs of your test plan.

#### - Write tests for your broader audience.

Think who may need to run your tests in the future and write them according to their technical level and their knowledge of the system. This will help you to give the test to a developer or even to a fellow tester from a different team and get a helping hand when you need it during your project. In some occasions it will be impossible to write a single test that suites all audiences and you may consider writing 2 separate versions of a single test.

#### - Not too long or too short.

Tests shouldn't be too long so that they take half a day to run, nor should they be so short that you need to create a Test Set of 20 tests to cover a single feature. Think about tests that between 45 and 90 minutes to run, and that cover a convenient area of the system in a single shoot.

#### - Don't write tests you don't need and run them as much as you can.

Writing a good test right the first time is difficult, this means that you may need to run it once or twice while still making changes to them before they are good enough (to be given to others or released to customers). Make sure you test-drive your tests before they go out the door 😊. Another important thing is to run your tests regularly or they will stop being relevant. This happens because the application will change, or the environment will be modified, or for a thousand different reasons that will require you to make small changes and to adapt your test to the current constraints. Making small changes is quick and easy, but making full overhauls is hard and may lead you to discard the test and write a new one altogether.

***Good test writing is an Art that takes time to learn  
and a Craftsmanship that takes perseverance to master***

## Biography



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

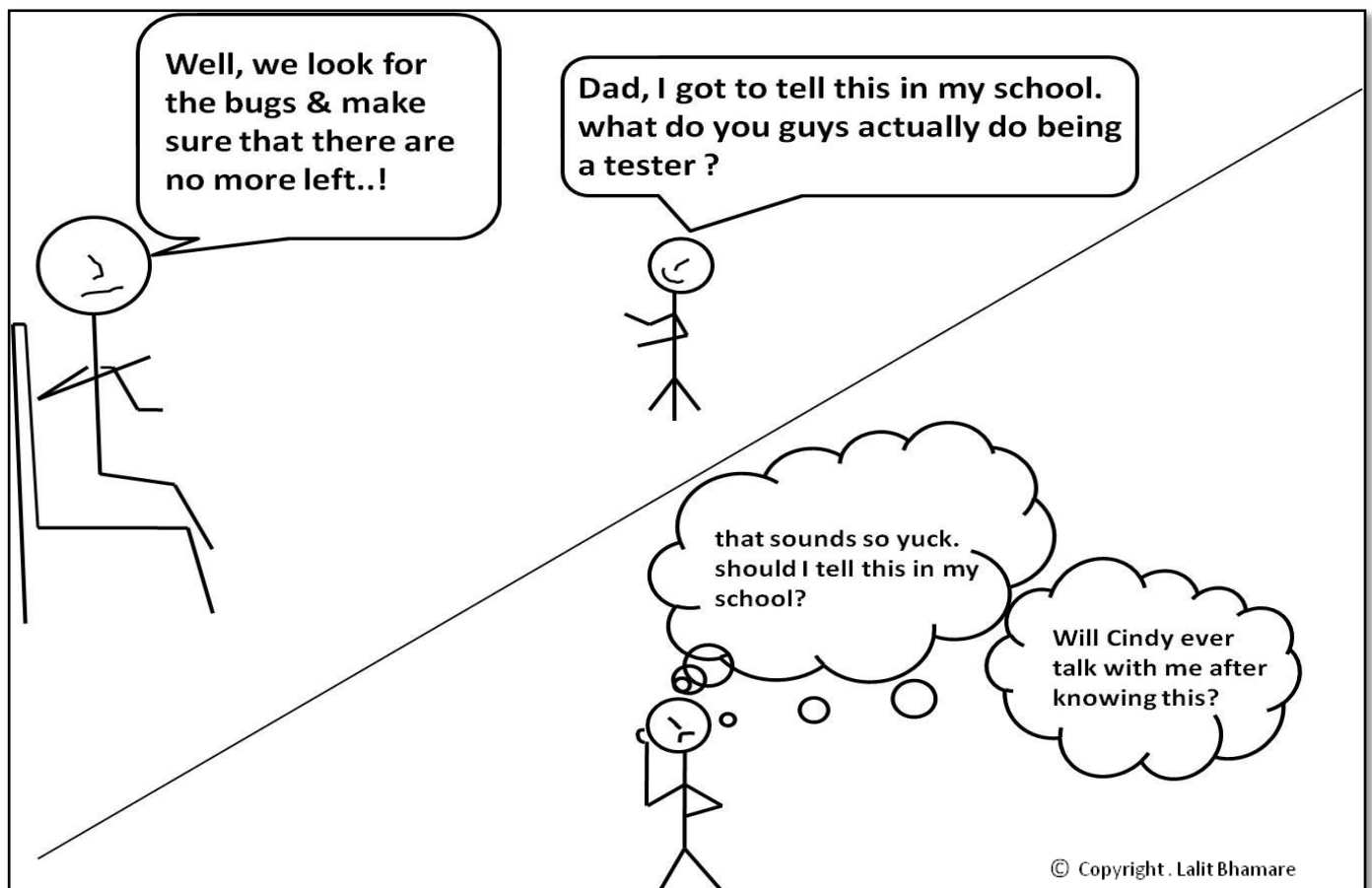
He's worked in companies ranging from small Internet Start-Ups and all the way to large multinational corporations, including Mercury Interactive (currently HP Software) where he managed the QA for TestDirector/Quality Center, QTP, WinRunner, and additional products in the Testing Area.

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](#)

## Tickle your Testing Bone..!! 😊





# THE SCRUM PRIMER PART 2

*by Pete Deemer, Gabrielle Benefield, Craig Larman, Bas Vodde*

Readers are encouraged to read the first part of this series in **Feb'11 Issue** of "Tea-time with Testers" - Editor

## Starting Scrum

The first step in Scrum is for the Product Owner to articulate the product vision. Eventually, this evolves into a refined and prioritized list of features called the **Product Backlog**. This backlog exists (and evolves) over the lifetime of the product; it is the product road map (Figure 2). At any point, the Product Backlog is the single, definitive view of "everything that could be done by the Team ever, in order of priority". Only a single Product Backlog exists; this means the Product Owner is required to make prioritization decisions across the entire spectrum, representing the interest of stakeholders and influenced by the team.

Item	Details (wiki URL)	Priority	Estimate of Value	Initial Estimate of Effort	New Estimates of Effort Remaining as of Sprint...					
					1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart (see UI sketches on wiki page)	...	1	7	5						
As a buyer, I want to remove a book in a shopping cart	...	2	6	2						
Improve transaction processing performance (see target performance metrics on wiki)	...	3	6	13						
Investigate solutions for speeding up credit card validation (see target performance metrics on wiki)	...	4	6	20						
Upgrade all servers to Apache 2.2.3	...	5	5	13						
Diagnose and fix the order processing script errors (bugzilla ID 14823)	...	6	2	3						
As a shopper, I want to create and save a wish list	...	7	7	40						
As a shopper, I want to add or delete items on my wish list	...	8	4	20						

Figure 2. The Product Backlog

The Product Backlog includes a variety of items, primarily new customer features (“enable all users to place book in shopping cart”), but also engineering improvement goals (“rework the transaction processing module to make it scalable”), exploratory or research work (“investigate solutions for speeding up credit card validation”), and, possibly, known defects (“diagnose and fix the order processing script errors”), if there are only a few problems. (A system with many defects usually has a separate defect tracking system.) The Product Backlog can be articulated in any way that is clear and sustainable, though either Use Cases or “user stories” are often used to describe the Product Backlog items in terms of their value to the end user of the product.

The subset of the Product Backlog that is intended for the current release is known as the **Release Backlog**, and in general, this portion is the primary focus of the Product Owner.

The Product Backlog is continuously updated by the Product Owner to reflect changes in the needs of the customer, new ideas or insights, moves by the competition, technical hurdles that appear, and so forth. The Team provides the Product Owner with estimates of the effort required for each item on the Product Backlog. In addition, the Product Owner is responsible for assigning a business value estimate to each individual item. This is usually an unfamiliar practice for a Product Owner. As such, it is something a ScrumMaster may help the Product Owner learn to do. With these two estimates (effort and value) and perhaps with additional risk estimates, the Product Owner prioritizes the backlog (actually, usually just the Release Backlog subset) to maximize ROI (choosing items of high value with low effort) or secondarily, to reduce some major risk. As will be seen, these effort and value estimates may be refreshed each Sprint as people learn; consequently, this is a continuous re-prioritization activity as the Product Backlog is ever-evolving.

Scrum does not define techniques for expressing or prioritizing items in the Product Backlog and it does not define an estimation technique. A common technique is to estimate in terms of relative size (factoring in effort, complexity, and uncertainty) using a unit of “story points” or simply “points”.

Over time, a Team tracks how much work it can do each Sprint; for example, averaging 26 points per Sprint. With this information they can project a release date to complete all features or how many features can be completed by a fixed date, if the average continues and nothing changes. This average is called the “velocity” of the team. Velocity is expressed in the same units as the Product Backlog item size estimates.

The items in the Product Backlog can vary significantly in size or effort. Larger ones are broken into smaller items during the Product Backlog Refinement workshop or the Sprint Planning Meeting, and smaller ones may be consolidated. The Product Backlog items for the upcoming next several Sprints should be small and fine-grained enough that they are understood by the Team, enabling commitments made in the Sprint Planning meeting to be meaningful; this is called an “actionable” size. One of the myths about Scrum is that it prevents you from writing detailed specifications; in reality, it is up to the Product Owner and Team to decide how much detail is required, and this will vary from one backlog item to the next, depending on the insight of the Team, and other factors. State what is important in the least amount of space necessary – in other words, do not describe every possible detail of an item, just make clear what is necessary for it to be understood. Low priority items, far from being implemented and usually “coarse grained” or large, have less requirements details. High priority and fine-grained items that will soon be implemented tend to have more detail.

## Sprint Planning

At the beginning of each Sprint, the **Sprint Planning Meeting** takes place. It is divided into two distinct sub-meetings, the first of which is called **Sprint Planning Part One**.

In Sprint Planning Part One, the Product Owner and Team (with facilitation from the ScrumMaster) review the high-priority items in the Product Backlog that the Product Owner is interested in implementing this Sprint. They discuss the goals and context for these high priority items on the Product Backlog, providing the Team with insight into the Product Owner's thinking. The Product Owner and Team also review the "Definition of Done" (which was established earlier) that all items must meet, such as, "Done means coded to standards, reviewed, implemented with unit test-driven development (TDD), tested with 100 percent test automation, integrated, and documented." Part One focuses on understanding *what* the Product Owner wants. According to the rules of Scrum, at the end of Part One the (always busy) Product Owner may leave although they *must* be available (for example, by phone) during the next meeting. However, they are welcome to attend Part Two...

Sprint Planning Part Two focuses on detailed task planning for *how* to implement the items that the Team decides to take on. The Team selects the items from the Product Backlog they commit to complete by the end of the Sprint, starting at the top of the Product Backlog (in others words, starting with the items that are the highest priority for the Product Owner) and working down the list in order. This is a key practice in Scrum: The Team decides how much work it will commit to complete, rather than having it assigned to them by the Product Owner. This makes for a more reliable commitment because the Team is making it based on its own analysis and planning, rather than having it decided by someone else. While the Product Owner does not have control over how much the Team commits to, he or she knows that the items the Team is committing to are drawn from the top of the Product Backlog – in other words, the items that he or she has rated as most important. The Team has the ability to lobby for items from further down the list; this usually happens when the Team and Product Owner realise that something of lower priority fits easily and appropriately with the high priority items.

The Sprint Planning Meeting will often last a number of hours, but no more than eight hours for a four-week Sprint – the Team is making a serious commitment to complete the work, and this commitment requires careful thought to be successful. The Team will probably begin the Sprint Planning Part Two by estimating how much time each member has for Sprint-related work – in other words, their average workday minus the time they spend attending meetings, doing email, taking lunch breaks, and so on. For most people this works out to 4-6 hours of time per day available for Sprint-related work. This is the team's capacity for the upcoming Sprint. See Figure 3.

<b>Sprint Length</b>	2 weeks		
<b>Workdays during Sprint</b>	8 days		

<b>Team Member</b>	<b>Available Days During Sprint*</b>	<b>Available Hours per Day</b>	<b>Total Available Hours</b>
Tracy	8	4	32
Sanjay	7	5	35
Phillip	8	4	32
Jing	6	5	30

\* Net of vacation and other days out of office

*Figure 3. Estimating Available Hours*



Once the capacity is determined, the Team figures out how many Product Backlog items they can complete in that time, and how they will go about completing them. This often starts with a design discussion at a whiteboard. Once the overall design is understood, the Team decomposes the Product Backlog items into work. The Team starts with the first item on the Product Backlog – in other words, the Product Owner’s highest priority item – and working together, breaks it down into individual tasks, which are recorded in a document called the **Sprint Backlog** (Figure 4). As mentioned, the Product Owner must be available during Part Two (such as via the phone) so that clarification is possible. The Team will move sequentially down the Product Backlog in this way, until it’s used up all its estimated capacity. At the end of the meeting, the Team will have produced a list of all the tasks with estimates (typically in hours or fractions of a day).

Scrum encourages multi-skilled workers, rather than only “working to job title” such as a “tester” only doing testing. In other words, Team members “go to where the work is” and help out as possible. If there are many testing tasks, then all Team members may help. This does not imply that everyone is a generalist; no doubt some people are especially skilled in testing (and so on) but Team members work together and learn new skills from each other. Consequently, during task generation and estimation in Sprint Planning, it is not necessary –nor appropriate – for people to volunteer for all the tasks “they can do best.” Rather, it is better to only volunteer for one task at a time, when it is time to pick up a new task, and to consider choosing tasks that will on purpose involve learning (perhaps by pair work with a specialist).

All that said, there are rare times when John may do a particular task because it would take far too long or be impossible for others to learn – perhaps John is the only person with any artistic skill to draw pictures. Other Team members could not draw a “stick man” if their life depended on it. In this rare case – and if it is not rare and not getting rarer as the Team learns, there is something wrong – it may be necessary to ask if the total planned drawing tasks that must be done by John are feasible within the short Sprint.

Many Teams also make use of a visual task-tracking tool, in the form of a wall-sized task board where tasks (written on Post-It Notes) migrate during the Sprint across columns labeled “Not Yet Started,” “In Progress,” and “Completed.” See Figure 5.

One of the pillars of Scrum is that once the Team makes its commitment, any additions or changes must be deferred until the next Sprint. This means that if halfway through the Sprint the Product Owner decides there is a new item he or she would like the Team to work on, he cannot make the change until the start of the next Sprint. If an external circumstance appears that significantly changes priorities, and means the Team would be wasting its time if it continued working, the Product Owner or the Team can terminate the Sprint. The Team stops, and a new Sprint Planning meeting initiates a new Sprint. The disruption of doing this is usually great; this serves as a disincentive for the Product Owner or Team to resort to this dramatic decision.

There is a powerful, positive influence that comes from the Team being protected from changing goals during the Sprint. First, the Team gets to work knowing with absolute certainty that its commitments will not change, that reinforces the Team’s focus on ensuring completion. Second, it disciplines the Product Owner into really thinking through the items he or she prioritizes on the Product Backlog and offers to the Team for the Sprint.

			New Estimates of Effort Remaining as of Day...						
Product Backlog Item	Sprint Task	Volunteer	Initial Estimate of Effort	1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart	modify database		5						
	create webpage (UI)		8						
	create webpage (Javascript logic)		13						
	write automated acceptance tests		13						
	update buyer help webpage		3						
	...								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5						
	complete machine order for pRank		8						
	change DCP and reader to use pRank http API		13						

*Figure 4. Sprint Backlog*



*Figure 5. Visual Management - Sprint Backlog tasks on the wall*

By following these Scrum rules the Product Owner gains two things. First, he or she has the confidence of knowing the Team has made a commitment to complete a realistic and clear set of work it has chosen. Over time a Team can become quite skilled at choosing and delivering on a realistic commitment. Second, the Product Owner gets to make whatever changes he or she likes to the Product Backlog before the start of the next Sprint. At that point, additions, deletions, modifications, and re-prioritizations are all possible and acceptable. While the Product Owner is not able to make changes to the selected items under development during the current Sprint, he or she is only one Sprint's duration or less away from making any changes they wish. Gone is the stigma around change – change of direction, change of requirements, or just plain changing your mind – and it may be for this reason that Product Owners are usually as enthusiastic about Scrum as anyone.

## Daily Scrum

Once the Sprint has started, the Team engages in another of the key Scrum practices: **The Daily Scrum**. This is a short (15 minutes or less) meeting that happens every workday at an appointed time. Everyone on the Team attends. To keep it brief, it is recommended that everyone remain standing. It is the Team's opportunity to synchronize their work and report to each other on obstacles. In the Daily Scrum, one by one, each member of the Team reports three (and only three) things to the other members of the Team: (1) What they were able to get done since the last meeting; (2) what they are planning to finish by the next meeting; and (3) any blocks or impediments that are in their way. Note that the Daily Scrum is not a status meeting to report to a manager; it is a time for a self-organizing Team to share with each other what is going on, to help them coordinate. Someone makes note of the blocks, and the ScrumMaster, is responsible to help Team members resolve them. There is no discussion during the Daily Scrum, only reporting answers to the three questions; if discussion is required it takes place immediately after the Daily Scrum in a follow-up meeting, although in Scrum no one is required to attend this. This follow-up meeting is a common event where the Team adapts to the information they heard in the Daily Scrum: in other words, another inspect and adapt cycle. It is generally recommended not to have managers or others in positions of perceived authority attend the Daily Scrum. This risks making the Team feel "monitored" – under pressure to report major progress every day (an unrealistic expectation), and inhibited about reporting problems – and it tends to undermine the Team's self-management, and invite micromanagement. It would be more useful for a stakeholder to instead reach out to the Team following the meeting, and offer to help with any blocks that are slowing the Team's progress **(To be continued in next issue...)**

## Biography



**Pete Deemer** is a founder of GoodAgile, and co-founder of the Scrum Training Institute.

Pete is an honors graduate of Harvard University, and has spent the last 22 years leading teams building products and services at global companies. Most recently he served as Vice President of Product Development for Yahoo!, where he led Yahoo's global adoption of Scrum, which grew to over 2000 developers worldwide during his tenure.

He can be reached at his mail id – [petedeemer@scrumtraininginstitute.com](mailto:petedeemer@scrumtraininginstitute.com)



**Gabrielle Benfield** is a Certified Scrum Trainer based in London offering classes in the United Kingdom and Europe. Gabrielle has over 18 years experience building enterprise software and web products at global companies and is a founder of the Scrum Training Institute, with Jeff Sutherland, the co-creator of Scrum, Pete Deemer and Jens Ostergaard.

Gabrielle works with clients from diverse industries including banking, telecommunications, and internet.



**Craig Larman** works as the lead coach of lean product development adoption at Xerox, and serves as a consultant for large-scale Scrum and enterprise agile adoption at Nokia and Siemens Networks (now, NSN), at Statoil and Kongsberg Maritim and Cisco-Tandberg (in Norway), at Alcatel-Lucent, and at Schlumberger and UBS, among many other clients.

He can be reached at his mail id - [craig@craiglarman.com](mailto:craig@craiglarman.com)



**Bas Vodde** is originally from Holland, however he has lived in China, Finland, China again and currently lives in Singapore. He led the Agile transformation project at Nokia Networks and later Nokia Siemens Networks, and currently works for his own company called Odd-e. He's been working with several large products and several large company change projects.

Bas can be reached at his website [www.odd-e.com](http://www.odd-e.com)





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](mailto:teatimewithtesters@gmail.com).

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

# T ' Talks



*T. Ashok exclusively on software testing*

## De-ticking a dog – What can we learn from this?

**I** had a beautiful and adorable dog – a black cocker spaniel (“Taggy”). A wonderful medium-sized dog with lovely dropping ears, she loved to be in the midst of us. Being a house-dog, I had a big challenge in ensuring that she had no ticks/fleas.



### The REAL bugs - Ticks and fleas

A quick tour of the insect world now...



Ticks are wingless creatures that live exclusively on the blood of animals for three of the four stages of their life cycle. They are equipped with an apparatus called Haller’s organ which senses heat, carbon dioxide and other stimuli to allow the ticks to locate the presence of an animal food source. Once found, they crawl on and embed their mouth parts into the animal’s skin and proceed to suck up its blood. Ticks suck the blood and become fatter and fatter and then can fall off the dog. Ticks congregate in colonies and are generally sluggish.

Adult fleas are wingless insects, generally smaller than a sesame seed, who feed on the blood of animals. Their proportionately enlarged back pair of legs gives them an extraordinary jumping ability. Did you know that if fleas were the size of humans, they would be able to jump over the St. Paul's Cathedral in London six hundred times for three days! <http://www.insectainspecta.com/fleas/cat/jumping.html> . Hanging on to your pet's fur with their claws, their needle-like mouth parts bite through the skin to suck up blood. Fleas are extremely agile, they move around.



Removing these bugs from a dog, particularly a hairy one is indeed a very daunting task. You never really know the number of bugs present and it's always a challenge to know if you did indeed do a good job. As a test practitioner, I was curious to learn from this activity to improve my skills to detect "software bugs".

### **Where did I look for the "bugs"?**

I discovered that the bugs were not uniformly distributed, but did not seem random either. I understood that the bugs like to congregate between the digits of the feet, the underbelly near the legs, behind the ears, around the neck, in areas with higher density of hair and also areas that were warm and clammy.

I also understood that the neighborhood of areas with dull hair, patchy skin, itchy areas were good candidates to examine for bugs.

### **How do I detect (& remove) these bugs?**

The typical ways are:

1. Periodic shots to prevent insect multiplication and killing those that are present.
2. Using a special flea dog collar to repel these insects.
3. Periodic brushing of the dogs (typically at least once daily) to brush the coat to remove these.
4. Parting the coat, manually looking for them and physically removing them.

During my experiments with the dog, I discovered a couple of interesting techniques to de-tick.

1. One of these was parting the hair in the opposite direction allowing me to see the skin and therefore allowing a better visibility to the presence of bugs.
2. When I gave the dog a bath, I discovered that the water on the dog made all the hairs stick and the bugs stood out like sore thumbs that I could manually shake out.

These were interesting to me as these seemed to me interesting techniques that made the "bug stand out"

## **What are the sources (causes) of these bugs?**

The typical reasons for the presence of these bugs are:

1. Interacting with other flea-infested dogs and picking the bugs from them.
2. Dog walking in areas that could have these bugs from other stray dogs. My dog was especially susceptible to pick up these due to her long droopy ears.
3. Inappropriate diet that makes the skin and the blood attractive for the bugs.

## **How do I know if I had done good job of “bug removal”?**

Some of the observations that could indicate a good “bug removal” are a shiny coat, less itching by the dog, no bugs found on the floor. These are in addition to the ensuring a lower bug density during the later cycles of the physical bug removal.

## **What can we learn from this story?**

That the act of uncovering defects in software/system software requires an intelligent examination of knowing what defects to look for, why these defects may be present, where to look for, how to detect these efficiently, and what information to seek to aid the execution of the various necessary activities.

In the case of “how to uncover bugs”, the dog experience shows that certain “bugs” are better detected/prevented via “periodic tick & flea injections” (i.e. internal examination) while some are better detected/prevented by “flea collar” (i.e. external examination).

Diligent observation of behavior (i.e. symptoms ‘like itching’), being sensitive to the context (‘areas of dog walking’, ‘stray dogs’, ‘diet’) is very important to effective testing.

That it is useful to understand where the potential defects may be present i.e. potential areas by careful observation/study.

That it is important to have rapid systematic scrubbing (‘daily brushing’) to lower the probabilities of defect multiplication over time.

That examination of end user results (‘less itching by dog’, ‘shiny coat’) are truer indicators of cleanliness of software in addition to the types and number of defects caught.



Interesting thoughts...

***Like the periodic injections given to the dog to prevent or detect the bugs, do we have something that we could inject the software with, to prevent/detect software defects? Could one of the analogies be "assertions"?***

***Like the water on the coat that made all the hairs stick together and therefore expose the bugs, what can we "coat the software with" so that the defects stick a better? Could the "memory leak detection" where we lace the code to surface the "memory leak issues" be an analogy?***

***What is the equivalent to "the act of parting the hair in the reverse direction to expose the bugs" in testing a software/system?***

#### **End note**

My dog has been my mirror; she helped me reflect on software testing.

I am in search for the "pill" and the "coat" that can "kill" and "shake" out defects.

## Biography



**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](mailto:ash@stagsoftware.com).

# finally the wait is over...

# Announcing...!

## Alap Patel as a “Smart Tester of The Month”

Dear Team,

Am working as a tester in TCS from last 1 year. I would like to thank you for this wonderful ezine and also for providing us the platform to showcase our skills. The ‘**Bug-Boss’s challenge**’ initiative is something I would want to have at my work-place too. It’s really a good way to encourage the fresh testers as well as those who want to sharpen their *Testing blades*. ☺

Regards,

Alap Patel (Mumbai, India)



## “Blogger of The Month” award goes to Sujit Dahibawkar



Hello All ,

I am Sujit P. Dahibawkar working as Test Engg. with Omnitech Infosolutions Ltd. for the past 8 months. I have been into this field for more than 3 years and have a groove towards performance testing. Initially, I was associated with Indiagames into manual testing of mobile applications.

That was the place where I developed a liking for testing and then decided to enhance my skills further.

Currently, I am gaining knowledge on LoadRunner – Performance Test Tool.

read Sujit's winning entry in our  
“**Tool Watch**” section

All can get in touch with me at [sujit.dahibawkar@gmail.com](mailto:sujit.dahibawkar@gmail.com) or on Facebook you can recognize me as Jeet P. Dahibawka

- Sujit Dahibawkar (Mumbai, India)

## Bug-Boss Challenge – Click here to know more

A black and white photograph of a man in a suit and glasses looking through binoculars. The image is framed by a thick black border. The text 'Tool Watch' is overlaid in a large, white, sans-serif font.

# Tool Watch

about various testing tool around

# WinTask

by **Sujit Dahibawkar**



## The professional automation tool for Windows™ and the internet

*Providing automation since 1997*

### About WinTask

WinTask is a scripting tool having the following features.

- Automate and schedule any repetitive task using this macro recorder for Internet and Windows applications, with its object-based capture replay user interactions feature.
- Create macros in any windows program or add a macro tool to your specific software, automate navigation within Web sites, fill Web forms automatically, extract Web data for instance to Excel, measure response times, test applications and Web pages, avoid repetitive data entry, copy data from one application or Web page to another.
- Record your actions, expand the scope of your macro by editing it and adding loops, branching statements, database access (300+ commands). You can even add a recovery proc in case an unexpected event occurs to ensure the robustness of your macro. Well-constructed helpfile and a lot of examples are included to get you started.
- NT/2000/XP/2003 Scheduler is a Service and you can start tasks even if the station or server is logged out. Scheduler is not supported under Vista.
- With such powerful features, you can use WinTask as a low-price automation testing tool: automate your functional testing process and regressions tests for your applications and Web sites.
- With WinTask, automate and schedule many activities that otherwise would have required hundreds of hours and as a result save thousands of dollars.

WinTask is used to produce automated test scripts with a .src extension.

Once created, WinTask scripts are compiled into 'Robot' files (with a .rob extension). It is these that are then executed, not the script files (.src) themselves.

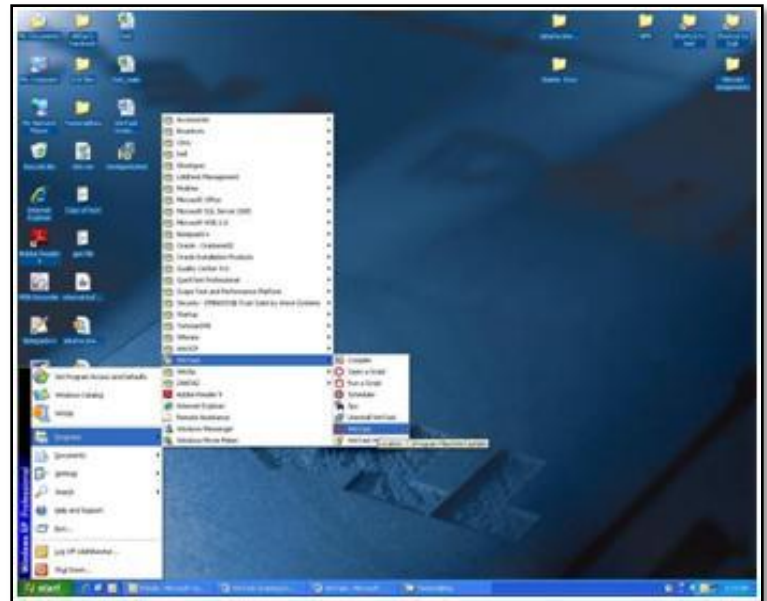
WinTask records at GUI level using 'Screen Scraping' methods, such as looking for particular Objects on a screen or GUI. A similar method to that adopted by QuickTest Professional

WinTask uses a completely unique scripting language, however it's significance is more or less like that of Vbscript.



## Starting WinTask Application

1. Goto Start → Programs → Wintask → Wintask
2. Your first script wizard window opens as shown.



This wizard basically prompts the user to record either an application or a website.

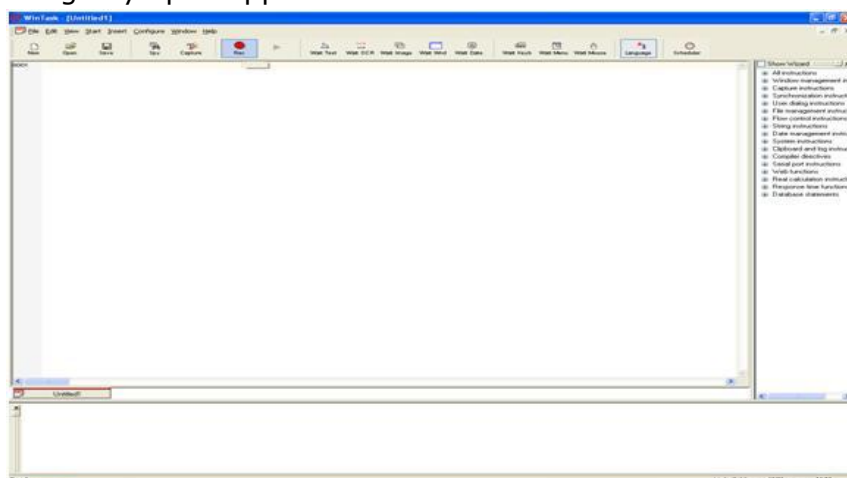
However I would recommend the user to cancel and continue recording from the main application window as it will be much easier to record from the main application window.

Once cancelled, the main WinTask application window should open as shown.

## Recording a Notepad Application in Wintask

When user decides to start a new recording, the 'Record' button at the top of the window is clicked. Please prefer this method.

Another alternate method is: Goto Start → Recording or CTRL+R. However this alternate method would directly start recording any open application.



On clicking the 'Record' button a dialog box is displayed that prompts the user to select from either of the 3 options: An Application or IE or Nothing.



**'An Application'** – This option allows any application on a machine to be started by supplying the path to an executable, the executable name and any parameters required by a specific application.

**'Internet Explorer'** – This option requires the URL for recording to be specified. IE will automatically be started and the specified web address will be displayed.

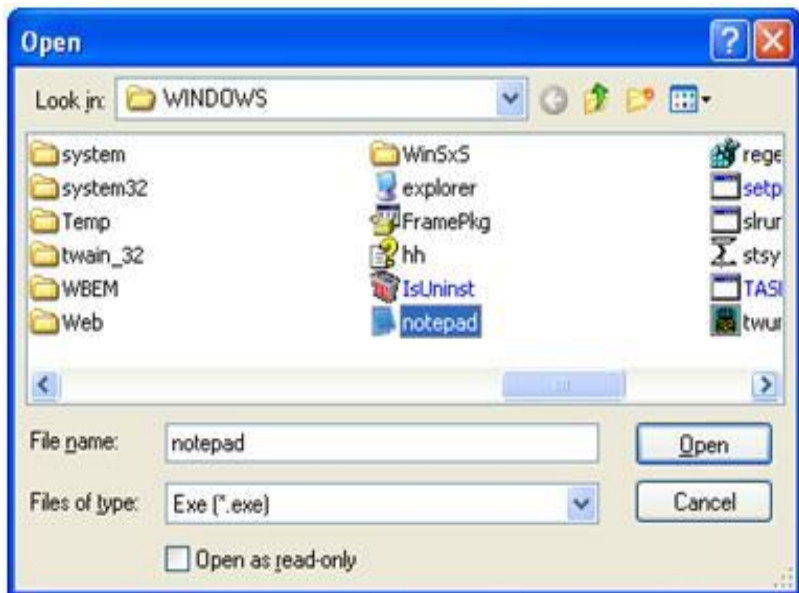
**'Nothing'** – This records any open application. This is particularly useful for adding in new steps to existing scripts to incorporate functionality or business process changes etc.

## Recording an Application:

### TASK 1: Automate a script to open an existing Notepad Application.

When the 'Browse' button under the 'Program' field is clicked the Open Dialog is displayed as shown in adjacent figure.

It is then possible to navigate to the executable of the Application Under Test, in this instance 'NOTEPAD.EXE' usually located in C:\WINDOWS



For this particular example covered in this document (Notepad), there are command line parameters that are passed to the application. Hence along with the full path and filename, the parameter file is required to start recording.

For this instance, create a parameter file by the name, say, '**text.txt**' enter some random text in it. Save it.

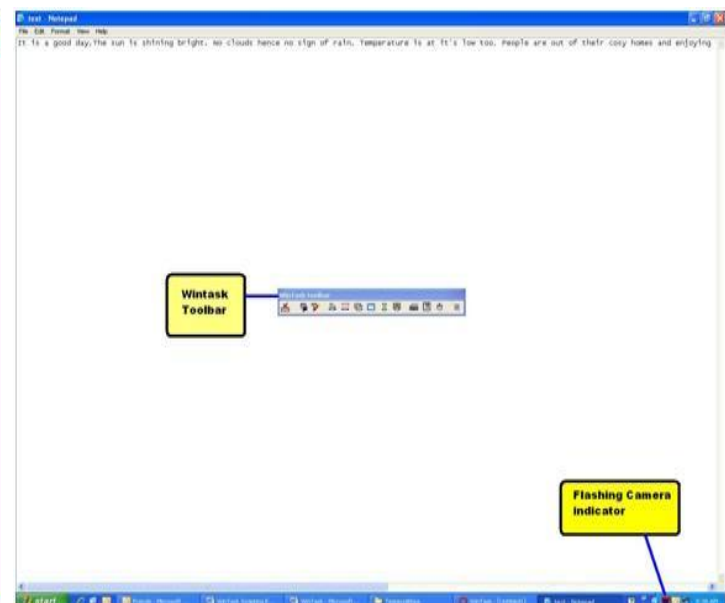
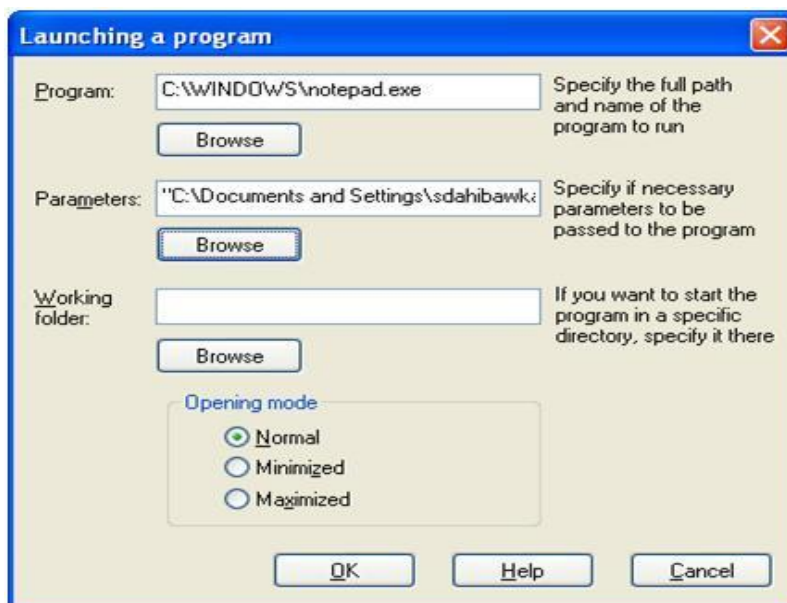
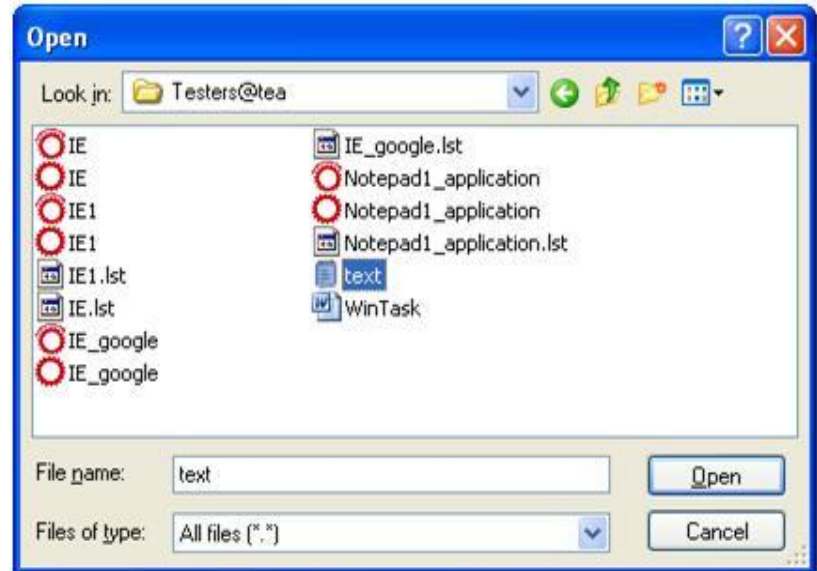
Now the path of this file is specified in the Parameters field as shown in adjacent figure:

Select the Opening mode i.e. Normal (Restored), Minimized, Maximized.

We have set it to Normal mode. Click Ok.

The Application Under Test ('text.txt') will open, the WinTask capture process will be started (indicated by the flashing camera icon in the system tray), and the WinTask toolbar will be displayed.

Any actions now carried out using the Notepad application will now be recorded.





## Script Creation

As with LoadRunner the process of creating a raw script is as simple as carrying out the process manually. However, the main difference is that where LoadRunner would be recording the backend send/receive requests, API calls etc. WinTask is recording the tasks carried out on the GUI itself.

For this reason, it is recommended that a logical approach is taken to the recording of a script, to make them as stable as possible in the first instance, for example, by using keyboard shortcuts or key presses where possible, as opposed to utilising the mouse. This provides increased stability of the raw scripts by negating issues such as the location on the AUT within the OS screen, or the specified screen resolution etc.

Once in the position detailed above, start recording a simple script such as that detailed below:


Start the script recording as detailed previously – Text.txt should open.

Select the entered text.

Select 'Format' followed by 'Wordwrap' (using the Alt shortcuts).

Make some changes to the formatting of the text e.g. font, size, etc. and accept these changes. Attempt to perform all these actions using only keyboard shortcuts i.e. tab, space, enter etc.

Either exit the Notepad application or close the created document, again utilising the Alt keyboard shortcuts.

Stop the recording by clicking the following icon from the WinTask Toolbar: 

The following code/recording performs similar actions to those detailed above. As you can see, all actions are performed using key presses.

```

/*****
Script developed by Mr. Sujit P. Dahibawkar. Script for recording a Windows Application (Notepad in this case).
Requisites for the below script to run is that the resolution of the machine where this is to be run should be the
same as the one on which it was recorded. Also a text file must be created and placed at the desired path so that
it can be passed as a parameter file when prompted during recording.

*****/

Introduces a delay of 1sec for each sendkey

*****/

#Sendkeysdelay = 10

*****/

Shell function executes a program (.exe, .com, .bat, .doc, .txt,...). In this case it is a text file
*****/
```

```
Shell("C:\WINDOWS\notepad.exe"+" "+Chr$(34)+"C:\Documents and  
Settings\sdahibawkar\Desktop\Testers@tea\text.txt"+Chr$(34),1)
```

```
/******
```

```
Maximizes the notepad window
```

```
*****/
```

```
UseWindow("NOTEPAD.EXE|Edit|text – Notepad|1",1)
```

```
SendKeys("<Alt >")
```

```
Sendkeys("<Down>")
```

```
Sendkeys("<Down>")
```

```
Sendkeys("<Down>")
```

```
Sendkeys("<Down>")
```

```
SendKeys("<Enter>")
```

```
/******
```

```
Select to format the text present in the notepad file
```

```
*****/
```

```
UseWindow("NOTEPAD.EXE|Notepad|text – Notepad",1)
```

```
ChooseMenu(Normal,"F&ormat|&Word Wrap")
```

```
ChooseMenu(Normal,"F&ormat|&Font...")
```

```
UseWindow("NOTEPAD.EXE|#32770|Font",1)
```

```
ChooseItem(Combo, "1", "Lucida Sans Unicode", single, left )
```

```
ChooseItem(Combo, "1", "Mangal", single, left )
```

```
ChooseItem(Combo, "1", "Marlett", single, left )
```

```
ChooseItem(Combo, "1", "Microsoft Sans Serif", single, left )
```

```
ChooseItem(Combo, "2", "Italic", single, left )
```

```
ChooseItem(Combo, "3", "11", single, left )
```

```
ChooseItem(Combo, "3", "12", single, left )
```

```
Click(Button,"OK")
```

**Readers are requested to continue reading this article on our blog site. Please click [here](#).**

## Our Testimonials

Your magazine seems to be pretty much inline with Context-Driven community.

Looks like you are off to a good start...!



- James Bach

Hi Lalit,

This is pretty impressive indeed!!!

Kudos & Keep the good work going, you are doing good for the community and this in time will be your biggest reward.



- Joel Monvelisky

"The Tea-time with Testers" first issue is an absolute blast with thought-provoking articles.

Thanks for sharing it :-)

- Aruna Venkatraman



I liked your first publication ... so I assume , others will too...!



- Jay Philips

Great test magazine, certainly looking forward to the next one.

- Marc Slijper





# You asked..

# We helped ...!

Hi Team,

Am facing some difficulties in using VB SCRIPT in QTP.  
Can you guys please help me?

- Nirmala Vare, Mumbai



Dear Nirmala,

Thank you for writing us. It will be our immense pleasure to help any tester as much as we can.

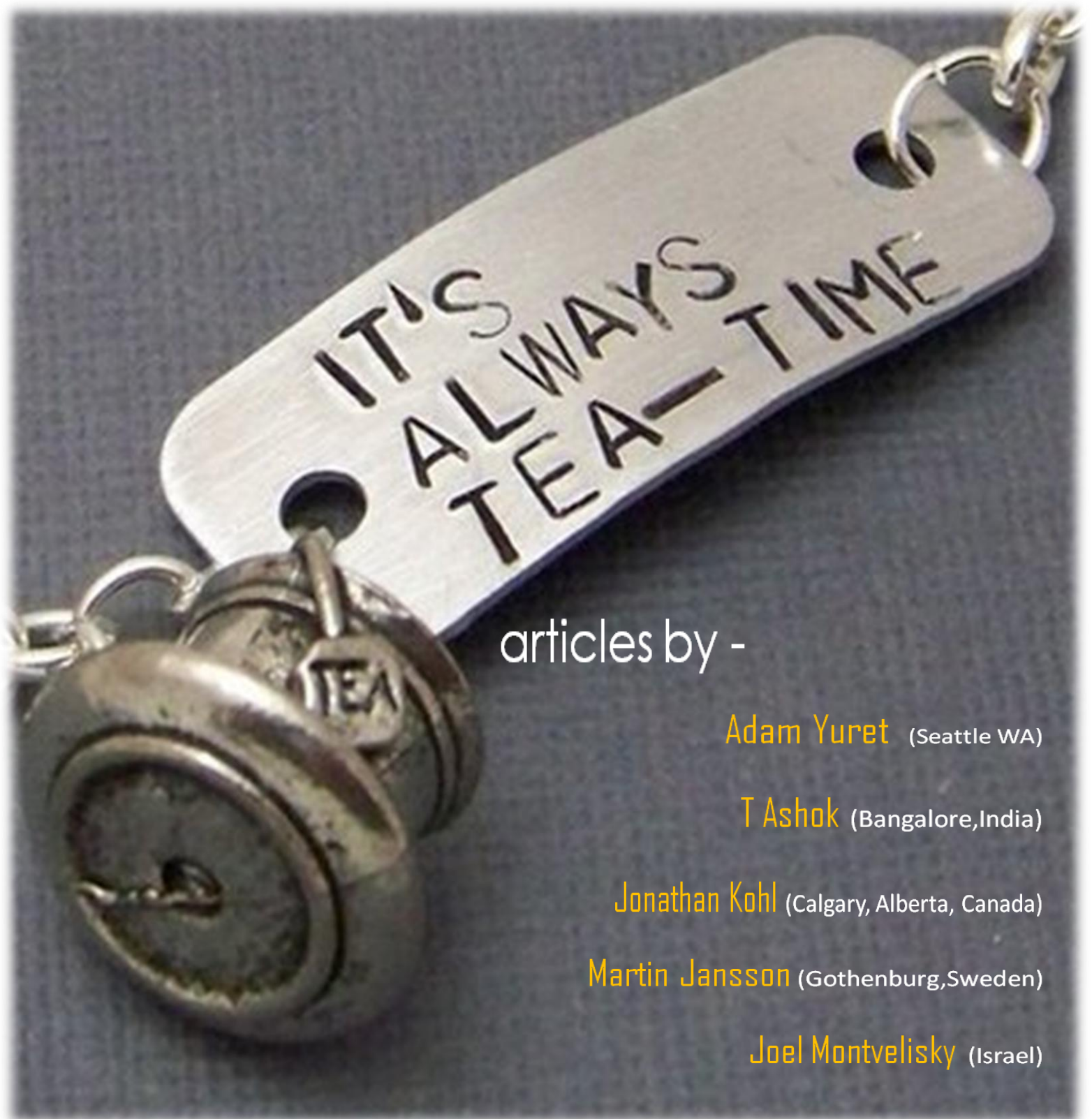
In order to help you out, *Chris* has written an article on **Learning VB Scripts for QTP**. You can read it on our blog-site by clicking **here**. Feel free to write us back if you have any specific queries. We shall try our best to address them. We have also mailed the PDF version of the same on your mail id. Happy learning.

- Editor

Feel free to write us your expectations.  
Help us to help you better.

We are just a mail away: [teatimeiwthtesters@gmail.com](mailto:teatimeiwthtesters@gmail.com)

# in ne>xt issue



articles by -

Adam Yuret (Seattle WA)

T Ashok (Bangalore, India)

Jonathan Kohl (Calgary, Alberta, Canada)

Martin Jansson (Gothenburg, Sweden)

Joel Montvelisky (Israel)



# our family

## Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

## Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover Page – Evening Flavors

## Core Team:

Kavitha Deepak (Bristol, United Kingdom)

Debjani Roy (Didcot, United Kingdom)



Kavitha



Debjani

## Mascot Design & Online Collaboration:

Juhi Verma (Mumbai, India)



## Tech -Team:

Subhodip Biswas (Mumbai, India)

Chris Philip (Mumbai, India)

Gautam Das (Mumbai, India)



Subhodip



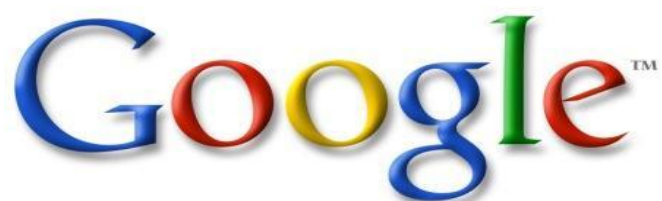
Chris



Gautam

*// 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](http://www.teatimewithtesters.com)

