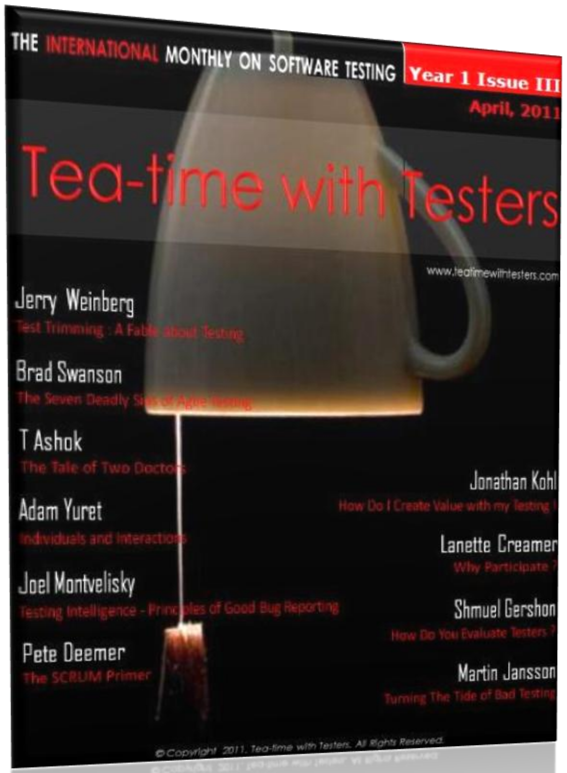
THE INTERNATIONAL MONTHLY ON SOFTWARE TESTING

Year 1 Issue III

April, 2011

Tea-time with Testers

www.teatimewithtesters.com


THE INTERNATIONAL MONTHLY ON SOFTWARE TESTING Year 1 Issue III
April, 2011

Tea-time with Testers

www.teatimewithtesters.com

Jerry Weinberg
Test Trimming : A Fable about Testing

Brad Swanson
The Seven Deadly Sins of Agile Testing

T Ashok
The Tale of Two Doctors

Adam Yuret
Individuals and Interactions

Joel Montvelisky
Testing Intelligence - Principles of Good Bug Reporting

Pete Deemer
The SCRUM Primer

Jonathan Kohl
How Do I Create Value with my Testing

Lanette Creamer
Why Participate ?

Shmuel Gershon
How Do You Evaluate Testers ?

Martin Jansson
Turning The Tide of Bad Testing

© Copyright 2011. Tea-time with Testers. All Rights Reserved.

Created and Published by

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

Editorial and Advertising Enquiries:

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

Editorial

Dear Readers,

I strongly believe that one of the best ways to learn software testing is to listen to the lessons learnt by others and learning from their experience.

All issues of *Tea-time with Testers* are special to me because what we strongly believe in is what we have offered you in these issues.

Sometimes people ask me, "What is so special with your magazine?" I politely ask them to read it once and check for themselves. The feedback I received is there in our Testimonial section.

It gives me an immense pleasure whenever I see testers sharing their experience via blogs, discussing over various topics of software testing. We all should call ourselves lucky for having such wonderful people and entire community around. I felt overwhelmed when I read Lanette Creamer's article *Why Participate* where she has poured her honest emotions and explained the need of participating in various event related to testing.

Jonathan's article *How do I create Value with My Testing* is a must read for all testers. Adam Yuret and Brad Swanson have written their brilliant thoughts around *Agile Testing*. As usual T Ashok has written yet another master piece with his brilliant imagination and experience. Do not forget to read *The Tale of two Doctors*.

Shmuel Gershon has cleverly expressed his thoughts on *Evaluating the Tester*. Martin Jansson's thoughts over Broken Window Theory in his *Turning the Tide of Bad Testing* will surely make you think deeper. Joel Montvelisky has continued to give his best through *Testing Intelligence*, the series you'd definitely recommend to Fresh Testers.

And now it's my honor to announce the new column *Tea & Testing with Jerry Weinberg*. Am sure, this is going to be the feast for you all.

The only reason for telling you about all these people is to let you know that we testers have really loving and brilliant *global testing community* around us which is always there to nourish our growth as a Tester and a wonderful human being too.

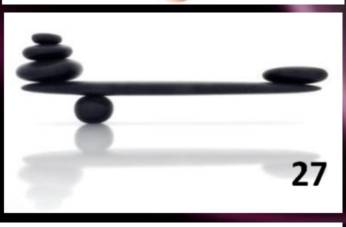
What all we need to do is just *to become part of it*. Enjoy Reading!

Yours Sincerely,

[illegible]

Lalitkumar Bhamare.

QuickLook



PractiTest

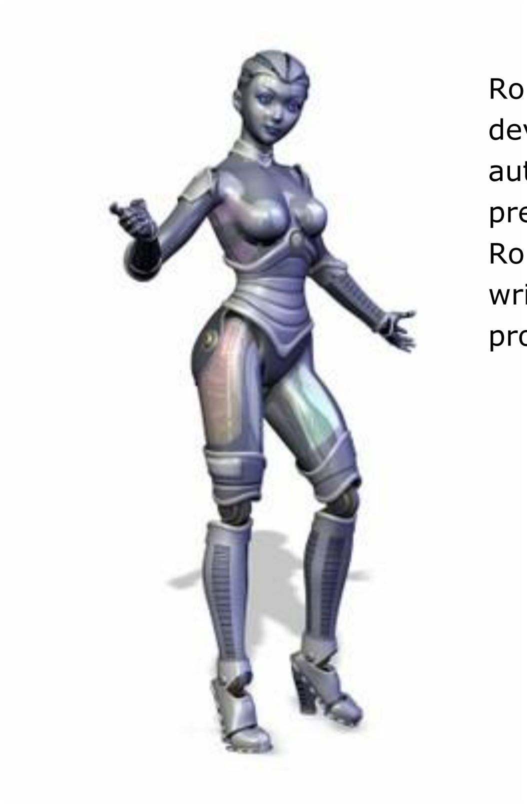
Cabot automates DTV testing

Digital device makers and pay-TV operators can reduce hardware and software testing time with Robotester, a new automated hardware and software testing framework for the DTV industry from DVB middleware vendor Cabot Communications.

Robotester's automated system performs key tests on DVB-enabled devices in 5 percent of the time it takes a person to do it manually, according to Cabot.

With speed to market being critical for digital device manufacturers, as well as satellite and cable operators, product testing often represents a time-consuming and resource-intensive phase of product development. A typical DTV system requires 20,000 to 30,000 tests, taking an average 100 to 150 days. With Robotester, manufacturers and operators can automate and perform 3000 tests in six hours nonstop, Cabot said.

CABOT

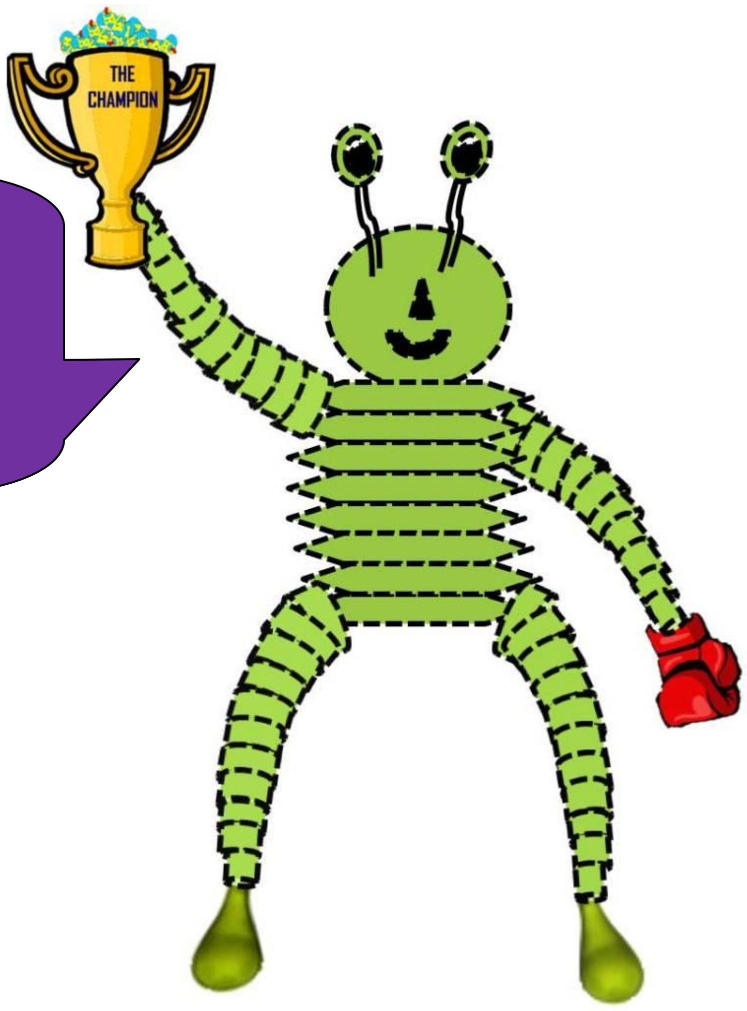


Robotester runs on a Windows PC and includes several peripheral devices used to simulate "real user" tests. Although the system is fully automated, the user can monitor the test progress at any time and is presented with a detailed structural log of the test results at completion. Robotester is customizable and includes an API that enables users to write and run their own tests with the free, object-oriented Python programming language.

-Broadcastengineering

Bug-Boss Challenge Results are out!

Scroll down to our **Announcements** Page !



"I am pleased to read Tea-time with Testers.

The Tester community has needed such a magazine for a long time, and you have done a creative and professional job of putting it together.

I'm looking forward to receiving future issues! "

- Jerry Weinberg

"I am pretty impressed with the quality and the content of the Tea-time with Testers.

I think it has a lot of practical knowledge, the finish-up is very professional and the overall feeling is of a project that's been running for a long time and not for only 2 publications.

It shows great achievements by a superb team! "

Cheers !

- Joel Montvelisky

Subscribe Right Away [FREE] for Future Issues

To Download our previous Issues click here

www.teatimewithtesters.com

Test Trimming: A Fable about Testing

Throughout my career, I've watched in dismay as one software manager after another falls into the trap of achieving delivery schedules by trimming tests. Some managers shortcut test work by skipping reviewing and unit testing in the middle of their project. Others pressure the testers to "test faster" at the end. And, most frequently, they just drop planned tests altogether, hoping they "get lucky."

I've written several essays about the dangers of test trimming, but nobody seems to understand, so I asked myself, "What am I doing wrong?" Perhaps I wasn't practicing what I was preaching. Perhaps I was trimming tests myself. Perhaps my writing needed more testing! So, I wrote a story about taking shortcuts and read it to my granddaughter, Camille.

Here's the story:

Rhubarb Cakes for the Queen of the Forest

Once upon a time, all of the animals in the forest were in an uproar. Calling Crow had just proclaimed that in two hours, the Queen of the Forest would arrive for a visit to choose a new Royal Baker.

"She's going to hold a baking contest," Calling Crow cawed. "And the winner will be named Royal Baker—and win a prize of 100 pieces of gold!"

"What do we have to bake for her?" barked Burly Bear. "Rhubarb cakes," Calling Crow cackled. "They're her very favorite dessert." Rapid Rabbit ran nervous circles around a rhododendron bush. "Rhubarb cakes? But the Queen is coming in just two hours, and my recipe for rhubarb cakes takes three hours."

"So does mine," complained Canny Coyote. Burly Bear sharpened his claws on the bark of an ancient aspen. "Mine, too." Prudence Porcupine popped up and headed for her kitchen. "Then I think I'd better get started, and not just stand around complaining?"

Each of the bakers pondered how they could make their rhubarb cake in two hours. "I've got it," thought Rapid Rabbit. "What takes the longest time is putting in a little sugar at a time, stirring for five minutes, and tasting to see if it's just right. I don't really have to test the sweetness a little at a time. I'll just throw in the right amount of sugar all at once, and that should save me an hour."

Burly Bear reasoned to himself, "I have the biggest oven, so I can put the cakes on the top shelf, in the very back where it gets super hot. And I can stoke the fire with lots and lots of apple wood because that burns hotter than any other wood. If I bake the cake at a higher temperature, I can save a lot of time and have it ready when the Queen arrives."

Canny Coyote didn't have such a big oven, but he figured, "if I just cut an hour off the baking time, the cake might be a little soft, but I'll put in lots of sugar. The cake will be so sweet that the Queen won't notice."

But Prudence Porcupine had a different way of thinking. "I know my rhubarb cakes are delicious, but if I take any shortcuts, I'm pretty sure the cake won't come out right. I'll just tell the Queen that my cake is going to be late, but that it will be worth waiting for."

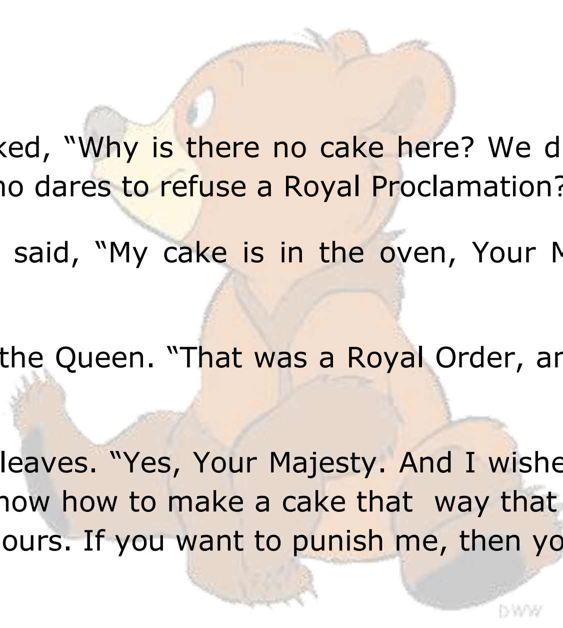
When the Queen arrived, Rapid and Burly and Canny all had their cakes on display in the clearing, and the Queen was invited to taste each cake in turn.

She took a bite of Rapid's cake and made an ugly face. "Yuck. This rhubarb cake is so bitter.

Why didn't you add more sugar?"

Then she turned to Burly's cake and asked, "Why is this one all burned and black? Well, maybe it's better on the inside." But when she tried to cut a slice, the burnt crust was just too hard to cut. Instantly, the Queen moved to Canny Coyote's table without even tasting the bear's cake.

She tried to cut a slice of the coyote cake. "This cake looks rather mushy. Oh, it's all gooey and runny when We try to cut into it. We think it would make Us sick if We put it in Our mouth." (Queens always call themselves "We" because they believe they are speaking for the entire nation.) The Queen stuck out her tongue at the cake and refused to taste it at all.

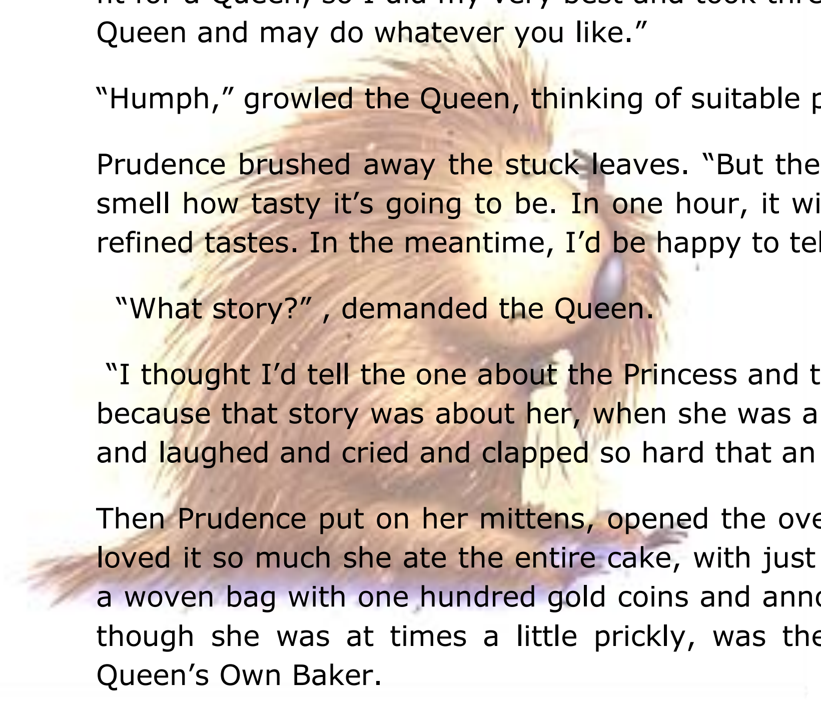


Finally, the Queen turned to Prudence's table and asked, "Why is there no cake here? We distinctly said that every baker was to make a rhubarb cake. Who dares to refuse a Royal Proclamation?"

Prudence stepped forward, bowed to the Queen and said, "My cake is in the oven, Your Majesty. It will be ready for your tasting in one hour."

"But We said the cake must be ready NOW " shouted the Queen. "That was a Royal Order, and cannot be disobeyed."

Prudence bowed so low her quills caught some fallen leaves. "Yes, Your Majesty. And I wished I could have made a rhubarb cake in two hours. But I don't know how to make a cake that way that would be fit for a Queen, so I did my very best and took three hours. If you want to punish me, then you are the Queen and may do whatever you like."



"Humph," growled the Queen, thinking of suitable punishments.

Prudence brushed away the stuck leaves. "But the cake is in the oven now, and you can just begin to smell how tasty it's going to be. In one hour, it will be fresh from the oven, and a fit dessert for your refined tastes. In the meantime, I'd be happy to tell you a story, to make the time pass more quickly."

"What story?" , demanded the Queen.

"I thought I'd tell the one about the Princess and the Pea," Prudence offered. This delighted the Queen because that story was about her, when she was a young princess. So the Queen listened to the story, and laughed and cried and clapped so hard that an hour passed by very quickly.

Then Prudence put on her mittens, opened the oven, and took out a perfect rhubarb cake. The Queen loved it so much she ate the entire cake, with just a small slice for Prudence. Then she gave Prudence a woven bag with one hundred gold coins and announced to the whole forest that Prudence Porcupine, though she was at times a little prickly, was the Best Baker in the Forest and now would be the Queen's Own Baker.

****

When I was finished reading, I asked Camille what was the lesson of the story.

She said, "Burly and Roger and Canny were not real bakers. They were just pretending because they wanted to win the prize, but they didn't know how to bake a real cake."

"Very good," I said. "But maybe they did know, but were afraid of the Queen."

"If you're afraid to do what you know is right, then you're not a real baker."

Camille, who was not yet five years old, understood this story perfectly, so it passed my test.

I wonder if forty-year-old software managers will be able to understand it ?

Biography

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

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

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

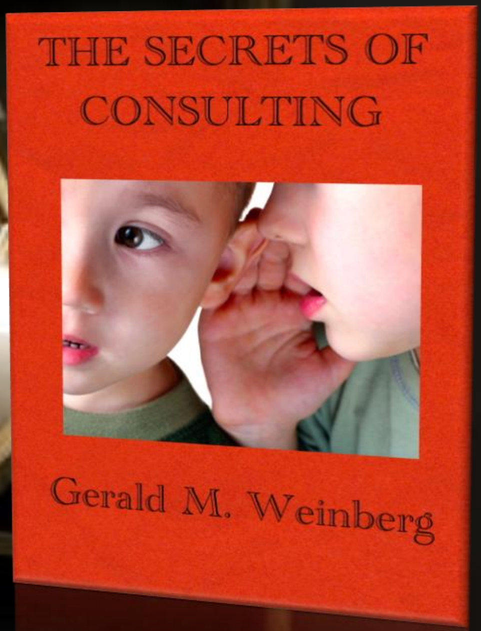
To know more about Gerald and his work, please visit his Official Website here .

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

The Secrets of Consulting is Jerry's one of the *most successful* and *recommended* book.

Its sample can be read online here .

To know more about Jerry's writing on software please click here .



Speaking Tester's Mind
- straight from the author's desk

Why Participate?

by Lanette Creamer

"Why can't you just go to work, and come home to your life? Why do you have to travel, speak at conferences, blog, and write articles? You know, I'd bet over half of all of the people who work in technology just do their work and then go home to their families and their lives."

I attended seven conferences last year, and spoke at six of them. I paid for my own travel expenses for most of them. I have a website without ads that I pay for out of what I earn professionally testing. There are assignments that other people take that pay for writing. I'm not very interested in the kind of writing that people pay for. The items that you see online, where you pick a buzzword that people are searching for and create content based on it. I'm not inspired to write for money. I write because I have something to share. There isn't anything wrong with being paid, and I think it is absolutely great to be able to write on demand, but I'm simply not driven to earn pay in that way. I have the luxury of writing only on topics that I'm interested in. Some of the best testers I know and admire also write, share, speak, and teach often at times, for free. I don't claim to have the right balance, because I haven't found my "sustainable pace" yet. I reduced my conference attendance this year, and seven was too many. I learned which ones I like the most, however, so that was helpful.

So why? Why can I continue to do this, when obviously it concerns the people who love me when I get the balance wrong? I remember testing when the only people I spoke to about it were nearly entirely in my company. At the time, I was able to travel around a great deal and learn most of what I needed to in one place. I'd write the occasional testing article for the internal newspaper, and participate when internal projects needed volunteers. Mostly, I'd go home when work was over and blog about personal things.

I need community, and our culture has a profound lack of it. I still had enough pride and company loyalty to keep all of my testing knowledge to myself. Afraid of all of the legal agreements I'd signed. I had not much to say except for how proud I was to work at the same place, and how much I enjoyed testing. But over time, testing has changed. The loyalty that I lavishly showered upon my company was slowly betrayed at deeper and deeper levels until the few left even could joke and dress as the "survivors" after each round of yearly layoffs. Knowing that the best testers I'd worked with could go down for being politically unpopular at the wrong time was a trust violation leaving the few testers left as nervous as a long tailed cat in a room full of rocking chairs. I was determined to make things better. No use complaining. Obviously, test leadership had failed to show the value of testing craftsmanship. They didn't understand. I could be a part of increasing the understanding.

Considering getting a company blog, I read through the paperwork and rules. I decided immediately that the company rules were in full conflict of creating an interesting blog. That is why so many company blogs are terrible, boring, and read like marketing material. I consider my candor one of my greatest assets. There are some who consider it a detriment, and those people generally aren't who I write for. They are still feeling safe and exempt from what large corporations do for the bottom line. They have in mind that what worked in 2000 is still working now, post recession. All that they need to rely on is their skill. They need no community. They need no real network, and no person even at work needs their cooperation more than a job competently done in mediocrity.

Everyone has a right to their opinion, but what makes me feel safe is learning new things, knowing new people, and gaining new knowledge. It doesn't just make me feel safe, it makes me feel energized, connected to people, creative, and alive. I have no church. I don't belong to a large extended family. I belong to the community of Software Testing, and I feel like I'm in good hands there. I find very few places where I feel understood, challenged, and in the company of smart, well intended, and helpful individuals trying to further something greater than themselves, even when there is no personal gain. Consider that the reason why there are no ads on my blog, there are no counters for tracking stats, and you'll see articles created without a charge. The question isn't why am I connected to the community, but in the past did I feel that it was best to sit back and watch, risking nothing, learning little? How much can you grow on your own in the next year? Five years? Decade? without practice, input, fellowship, and giving something of yourself. It can seem like there isn't enough time in the day, but we make time for what is important. I see how much others do without pay and I am inspired to do more, to be more, and to keep striving to find the right balance for me.

It's not that I'm giving up something for free. It's a fair trade.

More than fair to me.

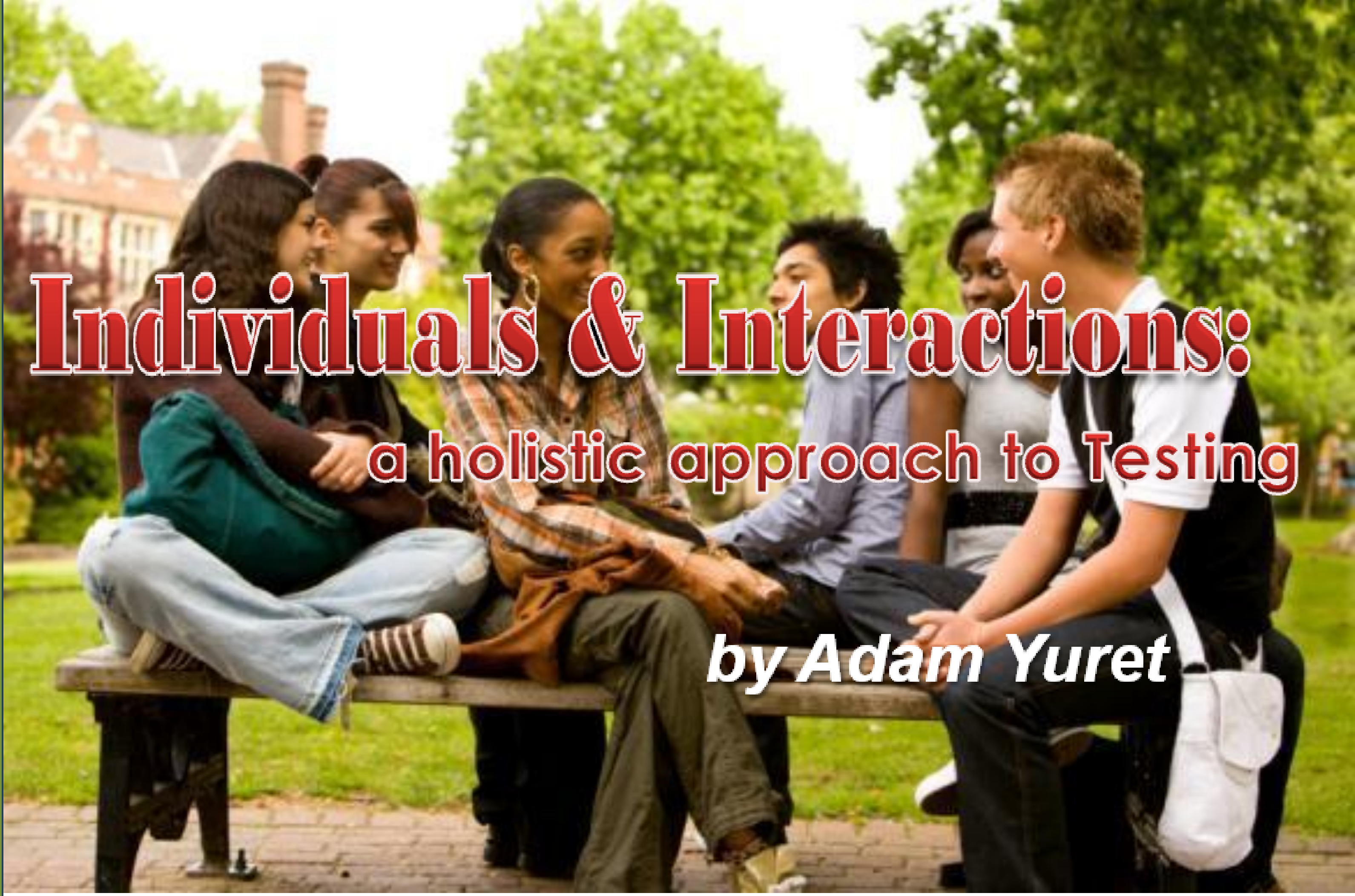
Biography

Lanette Creamer a.k.a. Testy Redhead is a test lead who is passionate about collaborative testing and is a practicing student of the context driven school. She has 12 years software industry experience, and has recently gone independent! Spark Quality LLC has one employee, Lanette!

Currently offering software testing, leadership, and coaching.

She has written 2 technical papers for PNSQC, 4 articles for testing publications, and often presents at software conferences. Check out her popular blog at **blog.testyredhead.com** .

Lanette can be found typing up a storm (up to the 140 character limit and beyond) on twitter **@lanettecream** .



Individuals & Interactions: a holistic approach to Testing

by Adam Yuret

One of the risks inherent in naming a team "Quality Assurance" is that , this team will be expected to enforce quality standards. By establishing such a team, some dangerous assumptions might be made that may result in undesired consequences.

Programmer: "I can focus on coding now that we have a testing team.

When programmers are the first and last line of defense against releasing bugs to the wild there may be greater attention paid to finding those bugs before they check them into the build or release them. By introducing a team that claims to "Assure Quality" we risk taking the onus off of programmers to check their work.

If the programmers are already practicing test driven development and testing their code as they work there is less risk in such an assumption.

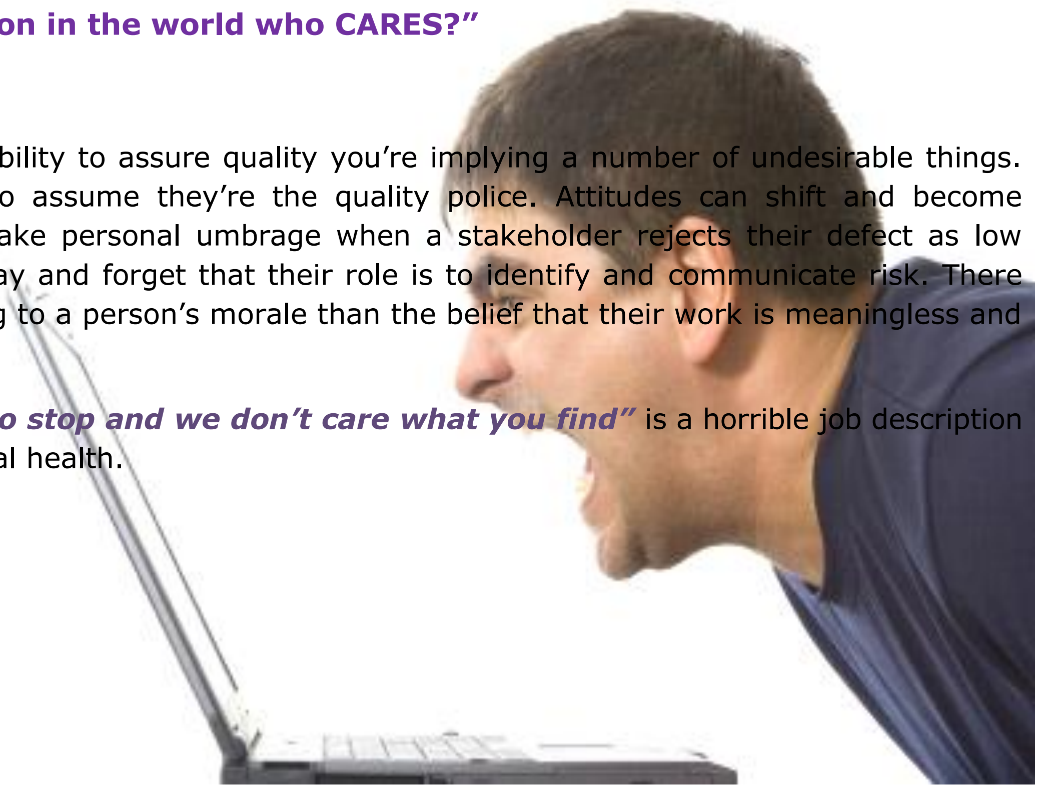
Programmers may be inclined to focus on engineering solutions to the requirements they've been handed and see the QA team as a safety valve.

In this case the QA team will spend much of their time sending bug reports up the line to development to fix basic fundamental flaws. The testers will not be using their skills to dig deep and find meaningful bugs through sapient testing. They'll become clerks, spending all of their time filing bug reports. It takes a lot more time to find and fix a bug when it's left the programmer's desk. Likewise the programmer will have moved on to other work and have to reset their thought process to dig into work they previously thought was finished to fix these issues. This lost programming effort will cause stories to slip and the stakeholder won't get all the business value they were promised during estimation. The consequences are shared across the whole team.

CRM: "I don't need to report when customer's have issues because I'm sure the QA team already knows about it."

By laying the responsibility for quality on a dedicated team you set up a situation where others may assume that customer issues are already being discovered by the testers. Maybe customer service reps would be more inclined to report issues to engineering if they felt like there was no buffer between programmers and customers.

Tester: "Am I the only person in the world who CARES?"



When give a team the responsibility to assure quality you're implying a number of undesirable things. It's very easy for the tester to assume they're the quality police. Attitudes can shift and become adversarial. The tester might take personal umbrage when a stakeholder rejects their defect as low priority. They may lose their way and forget that their role is to identify and communicate risk. There are fewer things more damaging to a person's morale than the belief that their work is meaningless and unending.

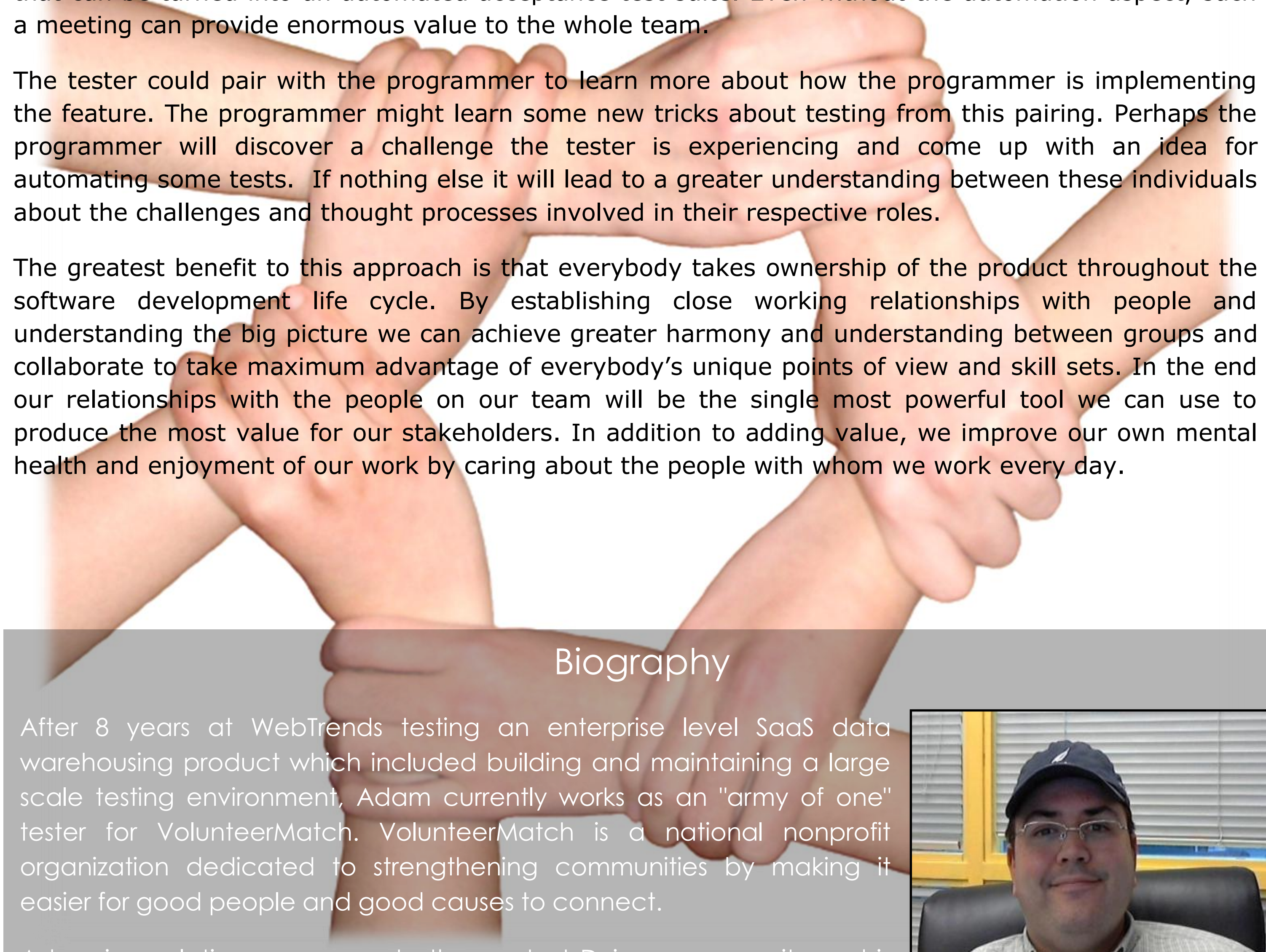
"Test this until we tell you to stop and we don't care what you find" is a horrible job description that is not conducive with mental health.

An Agile Solution

By collaborating with every member of the team we can eliminate many of these issues. Try holding brief requirements meetings (also known as story workshops) we can define at an early stage what is to be developed. Getting the programmer and tester together with the stakeholder in a room to flesh out requirements will lead to more detailed requirements. The tester's role in this meeting is to ask questions and outline boundaries with concrete examples provided by the stakeholder. The programmer can offer up his thoughts about the technical challenges of implementing such a feature. The stakeholder might see a shift in priorities as a result of the questions posed by the tester or the issues raised by the programmer. At the extreme end the tester might come away with a list of specifications that can be turned into an automated acceptance test suite. Even without the automation aspect, such a meeting can provide enormous value to the whole team.

The tester could pair with the programmer to learn more about how the programmer is implementing the feature. The programmer might learn some new tricks about testing from this pairing. Perhaps the programmer will discover a challenge the tester is experiencing and come up with an idea for automating some tests. If nothing else it will lead to a greater understanding between these individuals about the challenges and thought processes involved in their respective roles.

The greatest benefit to this approach is that everybody takes ownership of the product throughout the software development life cycle. By establishing close working relationships with people and understanding the big picture we can achieve greater harmony and understanding between groups and collaborate to take maximum advantage of everybody's unique points of view and skill sets. In the end our relationships with the people on our team will be the single most powerful tool we can use to produce the most value for our stakeholders. In addition to adding value, we improve our own mental health and enjoyment of our work by caring about the people with whom we work every day.



Biography

After 8 years at WebTrends testing an enterprise level SaaS data warehousing product which included building and maintaining a large scale testing environment, Adam currently works as an "army of one" tester for VolunteerMatch. VolunteerMatch is a national nonprofit organization dedicated to strengthening communities by making it easier for good people and good causes to connect.

Adam is a relative newcomer to the context Dr.iven community and is currently working to build a testing process for a project that is transitioning to an agile/scrum methodology.

Adam has recently made a presentation at QASIG (March 2011). Its video streaming can be viewed here. He can be contacted at adam.yuret@gmail.com or on twitter @AdamYuret.

7 Deadly sins of SE7EN Agile Testing

by Brad Swanson

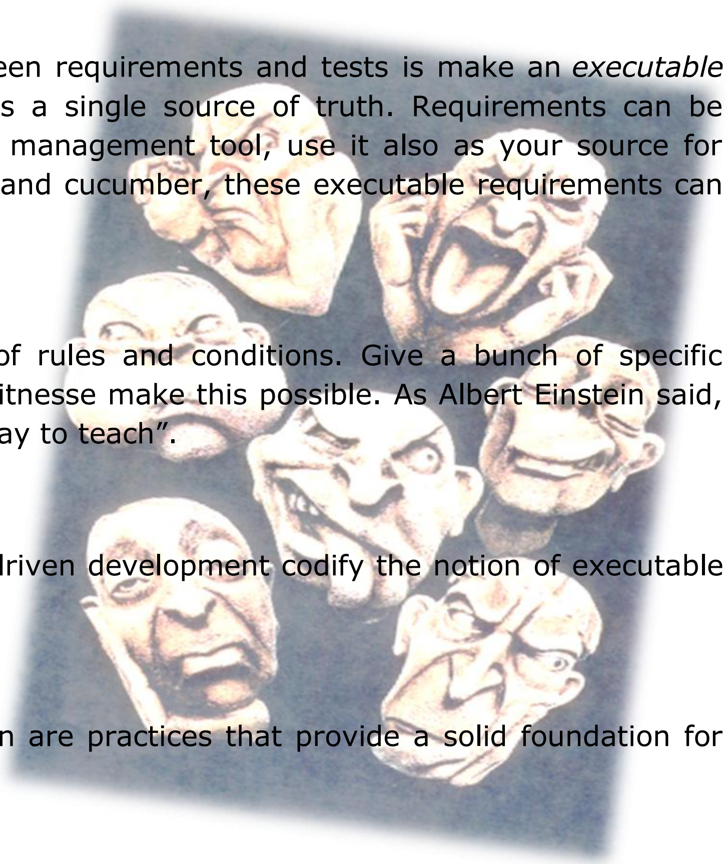
I had the honor of presenting at the **2011 SQuAD** conference on *The Seven Deadly Sins of Agile Testing*. It was a highly interactive session where participants gathered in small groups to identify *penances*, or ways they can avoid the seven sins. I provided a set of cards with ideas, and participants generated their own ideas as well. Below is a summary of the solutions the participants came up with for each of the seven sins.

Sin #7: Separation of Requirements and Tests

Sin #7 may not be quite deadly, but it definitely is a big drag on team effectiveness. When requirements are separated from tests, it's difficult to know which one is the source of truth, and maintaining the dreading traceability matrix is time consuming. Here are some ways to cleanse your soul from this sin.

Analysts & testers are best friends!

If we want to reduce the distance between requirements and tests, we need to reduce (or eliminate) the distance between the people responsible for them. Analysts can learn how to write tests, and testers can learn how to think like analysts. Closer collaboration between analysts, testers *and* programmers is a core value of agile methods and will help in many ways beyond this one.



Create an executable specification

The ultimate way to eliminate the separation between requirements and tests is make an *executable specification*. The tests *are* the spec. This provides a single source of truth. Requirements can be written in the form of tests. If you're using a test management tool, use it also as your source for requirements. Even better, with tools like fi tnesse and cucumber, these executable requirements can be automated.

Specify requirements by example

Don't limit requirements to abstract descriptions of rules and conditions. Give a bunch of specific examples to illustrate the requirements. Tools like fitnesse make this possible. As Albert Einstein said, "Example isn't another way to teach, it is the only way to teach".

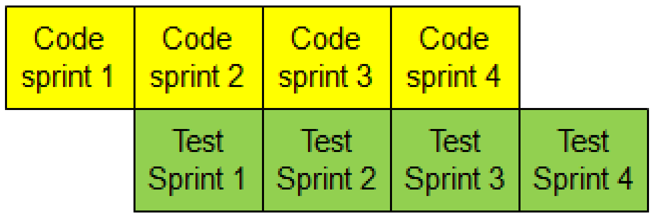
Practice ATDD and BDD

Acceptance test-driven development and behavior-driven development codify the notion of executable specifications.

Team commitment to TDD and CI

Test-Driven development and continuous integration are practices that provide a solid foundation for creating and executable specification.

Sin #6: Testing is one "sprint" behind coding



Scrum defines the output of a Sprint as a "potentially shippable product increment". If it's potentially shippable, it must be tested. Period. Here are some ways to atone for this sin.

Make user stories smaller

See Richard Lawrence's great post on **patterns for splitting user stories**.

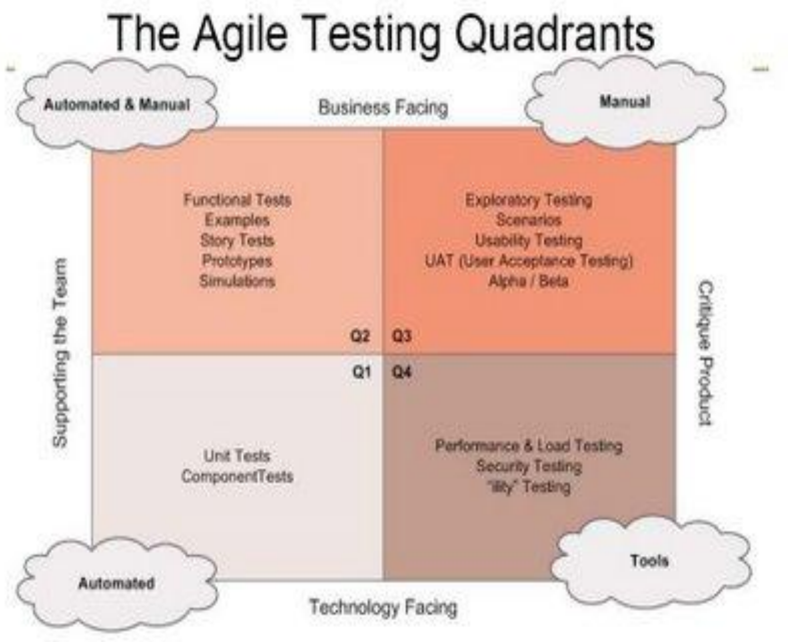
Here are some more ideas to bring testing and coding into the same sprint:

- The *Definition of Done* for backlog items includes testing
- Keep QA involved throughout the project – especially at the beginning of each sprint
- Whole-team responsibility for quality
- Closer collaboration between programmers and testers
- Create an executable specification (see Sin #7 above)

- Specify requirements by example
- Include testing effort in release planning
- Team commitment to TDD and CI
- Minimize the effort to setup and deploy test environments
- Programmers deliver working code to testers early in the sprint/iteration
- Co-located teams
- Pair development
- Persevere through challenges
- As a last resort, you may consider increasing the sprint length, but this option has many disadvantages also.
- Consider a Scrumban/kanban process that is flow-based, rather than iteration-based. Even so, strive to keep backlog items small!

Sin #5: Unbalanced testing quadrants

The **agile test quadrants** define different types of testing necessary on most projects: unit testing, acceptance testing, exploratory testing, and "property" testing (performance, security, etc.). Our participants identified a few ways to absolve a team of this sin.



- Balance the testing quadrants in your Definition of Done – at the user story and release level
- Categorize tests to analyze quadrant coverage, determine proper weighting of each
- Include testing in release planning
- Automate acceptance tests, allowing more time for other quadrants
- Keep QA involved throughout the project
- Exploratory testing included in acceptance testing with the customer
- Outsourcing or in-sourcing specialized testing (performance, -ility) earlier in project.

Sin #4: Ignoring Test Failures

Strong agile teams tend to have a lot of automated tests that run multiple times per day via continuous integration. All that effort might be for naught if you don't do anything when one of those tests fails! Pay your penance, as suggested below.

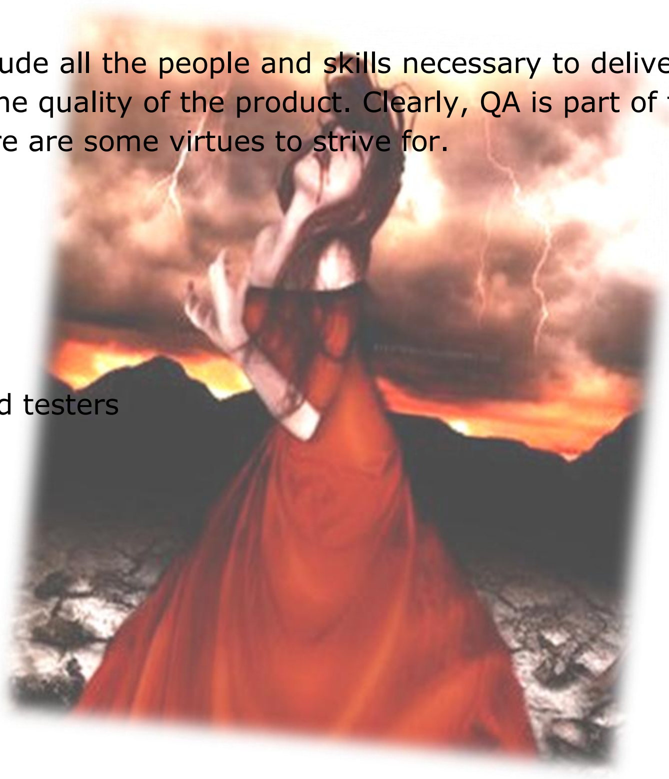
- "Stop the line" when tests fail
- The Definition of Done for backlog items includes testing
- Stakeholders participate in testing
- Invest in *robust* automated tests
- Incentives for clean check-ins. A little peer pressure or friendly competition can help establish a culture where clean check-ins and passing tests are the expectation of everyone.

Sin #3: Lack of Test-Driven Development (TDD) and Continuous Integration (CI)

TDD and CI are core practices from Extreme Programing and agile teams will quickly hit a brick wall without them. Here are some ways for a team to get on the path toward righteousness.

- Achieve an explicit team commitment to do TDD and CI
- Invest in legacy test automation
- Make automated tests robust
- Practice ATDD and BDD
- Keep metrics on important quality indicators and monitor trends; use these metrics to demonstrate ROI on TDD and CI. Typical measures include rate of CI build failures, test coverage, manual effort level per test cycle, escaped defect counts, and customer satisfaction.
- Prioritize test automation efforts to maximize ROI
- Stakeholders participate in testing
- "Stop the line" when tests fail
- Include testing in release planning
- INVEST in user stories; they should meet the INVEST criteria. Make them small and testable.

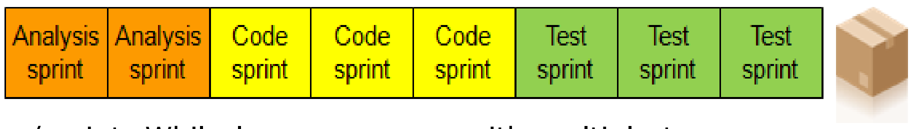
Sin #2: Separate QA team



Scrum teams are cross-functional, meaning they include all the people and skills necessary to deliver a working product. The whole team is responsible for the quality of the product. Clearly, QA is part of the development team, rather than a separate team. Here are some virtues to strive for.

- Cross functional teams include QA
- Keep QA involved at all times
- Whole-team quality responsibility
- Closer collaboration between programmers and testers
- Include testing in release planning
- Include QA activities in the product backlog
- Co-located teams
- Open Communication

Sin #1: Waterscrumming



The last and perhaps worst of the seven sins is being Agile in name only. High quality, end-to-end tested features are the output of every iteration/sprint. While large programs with multiple teams may very well need a *single*hardening sprint, multiple testing sprints is just another form of waterfall. Many of the penances listed for the other six sins also apply here.

- Acquire deeper knowledge of the agile practices. Understand the lean and agile principles behind the process.
- Do a pilot project. Be disciplined about all the practices and resist the temptation to modify the practices until you've first tried them "by the book" for a significant time.
- Include testing in release planning. Testing is an integral part of the team and process, not an afterthought.
- Include QA activities in the product backlog. Quadrants 3 and 4 in particular may require focused effort that is not directly related to individual features built during sprints.
- Cross-functional teams include QA
- Co-located teams
- Whole-team responsibility for quality

- The Definition of Done for backlog items includes testing
- Close collaboration between programmers and testers throughout
- Create an executable specification
- Specify requirements by example
- Pair development. In particular, pairing testers with programmers is a great way to cross-train and enable shorter cycles.
- Prioritize test automation efforts to maximize ROI on test automation.
- Invest in robust automated tests. Tests should be modular and robust in the face of changes in the system.
- Better project management. Traditional PMs need to understand agile principles & practices and may need coaching to implement agile practices effectively.
- Incentives for clean check-ins
- Test First! Practice TDD, ATDD and/or BDD
- Ask for forgiveness rather than permission. Can you take the initiative to apply agile practices within your sphere of control and then demonstrate the results, rather than asking for permission first?

I welcome input from my fellow sinners who have found their own paths to righteousness. Please write to me with your ideas for salvation!

Biography

Brad Swanson is a Certified Scrum Coach and Principal Consultant at Propero Solutions. He started his software career at age ten on the Apple IIe, and is now a Certified Scrum Coach (CSC), Certified Scrum Practitioner (CSP), and Certified Scrum Master (CSM) with 15 years of experience. He has deep experience with agile software development, starting with eXtreme Programming (XP) in 1999, and also Scrum, Lean and Kanban methods.

He is active in the Agile and Scrum communities as a Director of **Agile Denver**, speaker at agile community events, volunteer for Agile conferences and speaker at international conferences such as the Shanghai Scrum Gathering.

Brad can be contacted at **brad@properosolutions.com** or on Twitter **@bradswanson**

Connect to Audience That Matter

Advertise with Us

Contact: Lalitkumar +91-9960556841
Pratikkumar +91-9819013139
or
Mail to : teatimewithtesters@gmail.com

In the school of Testing
for your better learning & sharing experience

How do I create Value with my Testing !

by Jonathan Kohl

I had written an article for EuroStar about creating value with testing. In the article, I talk about getting feedback from stakeholders, but that isn't always easy or possible. One of the most important stakeholders on any project is *you*, so how do you go about satisfying yourself with your testing value?

The easiest way to get feedback is from other stakeholders. What does your manager think about your testing? How about the programmers, business analysts and customers (users) of your software?

The hard part with that answer is you may not be able to talk to all of those stakeholders. Or, they may not know what good testing looks like so they won't have answers that satisfy you. In some cases, the stakeholders around you may have such low expectations that their feedback might not help you at all. They may expect you to provide testing work that you might consider shoddy and negligent. In that case, you have to show them what great testing looks like. When that happens it's like graduating from a cheap box of wine to the good stuff. Once they've tasted the good stuff, it's hard for them to go back to expecting poor testing.

Even if you have good direction from other stakeholders, I recommend asking yourself some questions to help determine if you are creating value or not. This is hard to do, and will result in work for you over the long-term, much like personal growth endeavours.

Don't expect quick fixes, but if you work in these areas over time, you will see changes in your testing. Here are some things to think about:

- Is my testing work defensible? (Cem Kaner talks a lot about this.) Think of a court case. What would a jury think if you testified and described what you did as a tester and why. How did you determine priority? Why did you test some things and not test others? (100% complete testing is impossible, so you have to make decisions to optimize your work. Are those decisions well thought out, or more subconscious? What sorts of things might you be missing that you haven't thought of?)

- James Bach talks about how important it is to have thought out and varied *approaches* to testing. What kind of approach do I have to testing? Do I consciously choose to have a varied approach using as many models of coverage as I can to discover important information about the product? Do I make the best use of tools, testing techniques and management approaches that I can? Or do I just do what the programmers or someone else tells me to do?

- In the absence of getting real feedback from real people on my project, what would happen if a well-known consultant came to visit me? Could I answer their questions about why I chose to test this way? What kinds of holes might they spot in my thinking? Would they see weak spots? More importantly, would I be proud to have Cem Kaner or someone else I look up to see what I actually do? Have I used ideas from testing thought leaders in my work and found out what works well for me and what might not work so well? Could I communicate my work to an expert outsider clearly and thoughtfully? If so, what might they think?

- Do I adapt my test plans and strategies from project to project based on the risks and rewards our project environment has at a particular point in time, or do I just copy and paste what I did last time, and repeat the same thing over and over?

- Do I track down and find repeatable cases for important intermittent bugs, or do I just file them and forget about them?

- Do I feel energetic, creative and proud of my work as a tester, or do I just feel like I am doing the same boring things over and over and filling in paper work and forms to please a manager?

- Can I look at a released product and identify ways in which my testing has improved the product experience for our end users?

- Do others on my team feel better with me around? Do they miss me and my creative input when I am away, or do they welcome the break from my negativity? Do they request that I work with them on other projects?

- Is my testing service in demand? Am I the person team members come to when they need help solving a particular problem that I am really good at helping solve?

- Am I aware of other approaches to testing that challenge my favorites? Do I understand approaches that I may not favor or I may even dislike, or do I just dismiss anything unfamiliar and threatening out of hand? Do I have an open-mind and look to challenge my ideas in testing to help improve?

Biography

Jonathan Kohl is the founder and principal software consultant of Kohl Concepts, Inc.

Noted software testing thinker and strategist, Jonathan is a natural investigator on software projects.

In addition to assisting teams with testing, Jonathan helps companies define and implement their product vision; coaches practitioners as they develop software on teams, and works with leaders helping them define and implement their strategic vision.

Jonathan supports and contributes to Open Source testing tools. He is the founder and project lead on the Session Tester project, and was an original commiter and contributor to the Watir project.

Jonathan is also a popular author and speaker. His blog on software development and testing issues is one of the most well-read testing blogs in the industry.

Jonathan can be contacted at http://www.kohl.ca/.

- Am I learning about different ways I could improve my work? Am I aware of recent changes in testing techniques and tools? Do I know where to find information to learn from?
- Do I consistently try to do better than I did last time?

These are the kinds of questions I ask myself regularly.

I don't always have the best answers to my own questions, but as time goes on, I feel much more confident about both my own answers to those questions, and more importantly, the value I know my testing work provides.

How do you Evaluate Testers ?

by Shmuel Gershon

I frequent many of online forums. It's a great way to grow and learn, as the discussions are fantastic opportunities to challenge us with questions we don't face on our day to day work. I also enjoy the questions that I do encounter in my day to day, because when answering these (*or reading answers*) for a context different from mine, I discover new ways of thinking about matters I do out of routine.

In English, an interesting site is the Testing.StackExchange , Questions & Answers site. Unfortunately the site is less active than it could be, but there are good questions and very good answers in the list. And even some big names like Alan Page, Michael Bolton and BJ Rollison answer questions there, so it's a good place for you to ask yours.

I came across an interesting question popped at the Testing.StackExchange forum:

How can you evaluate the performance of a tester? How can you compare testers' efficiency? This is a question that entertains me once a year, when my company does employee evaluations, and my trial at answering it is in this post. Can you suggest other points of view? What else you recommend to consider? How do **you** evaluate testers?

The question goes similar to this:

If I assign one requirement to two testing engineers, test will come up with 10 test cases for the given requirement and the other one will come up with 15 test cases. Both of them affirm to cover all the scenarios with their tests

How can I decide which one is better? Apart from considering the minimum amount of test cases are there other factors to decide who the most efficient tester is?

My answer:

If you don't want to consider the minimum number of test cases, you can consider the one with the most number of test cases then… 😉

Jokes aside, trying to determine efficiency of testers **based on numbers like that** will lead you very far from the answer you look for. For instance, to be simplistic, both testers can be doing the exact same tests, just by dividing them differently in 10 or 15 cases. Or both testers can be doing a terrible job, but the mere numbers won't tell you that. The number of tests executed or planned does not show the contribution of a tester to a project. By counting test cases you are looking at something very superficial. The same will happen with any other measurable simple dimensions.

In order to decide which tester is better, we have to do it in **the same way we decide which of our friends is the best friend. It takes time and intimate acquaintance**. Which tester gives better and more useful information to managers? To programmers? To peers? Which one communicates better? Which one is funnier? Which one makes the job easier for the rest of the team? There is no one discipline where you can rate the testers fairly. One may find the most bugs, the other prevents most bugs, another may help the more peers, other has a better relationship with programmers. And the other one, that one in the corner that looks less efficient than all the other testers… well, that one is the one doing the support work so all the other testers can shine.

Think of all the dimensions where testers affect a product/project. You have to study subjectively the performance of the testers in **all** these dimensions in order to compare them. Testing skills, communication skills, teaching skills, writing skills, team skills, truth skills, learning skills…

Yes, it sounds difficult. Maybe because it **is** difficult but for any other arbitrary measurement you pick to make the task easy instead, you may as well randomly draw name slips from a hat.

While we are at that, do we really need to know which one is the **most** efficient? Does this comparison benefits our judgment and decision (*and consequently the team and product*)? Done wrongly, you may end up with a group of all non-efficient testers. Again, think of the similarities with the group of your friends: If you propose a clear-cut competition between your pals to determine which one is your best friend, some of them will stop being your friend just for proposing it, and others will become worse friends precisely for trying to answer the defined criteria.

If you have to rate, think about evaluating team mates individually rather than comparatively. May show you facets of your team you hadn't notice. A conversation with each tester about his own performance can do wonders, too.

And do you really care which one is more **efficient**? Would you prefer efficiency over honesty? Over willingness? Over results?

What I am saying is that there **is** a systematic way to tell how good a tester is. But it is not an easy "paint by numbers" method. In fact, including numbers is a step for the result to go awry.

The systematic method involves trust and respect, and like I wrote, it requires intimacy and subjectiveness (*don't know why people are afraid of sujectivity when you can't avoid it, just maybe mask it*).

The systematic method involves constant feedback and conversations about value. People usually know (*or can talk about*) if/how they can do the details better if they have a clear picture of their general situation.

Tom DeMarco has written in his **article** that "Most things that really matter—honor, dignity, discipline, personality, grace under pressure, values, ethics, resourcefulness, loyalty, humor, kindness—aren't measurable." These things are not measurable in numbers, but they are easy to feel.

Talking with the tester and with the team, you can get a sense of who provides value in his own eye and in the eyes of others. And this value can be of a kind that surprises you – if you set a specific defined criteria beforehand, you're sure miss these kinds.

Biography

Shmuel Gershon is a Testing Engineer at Intel Corporation working at the Intel vPro team.

He coaches testers and teams and presents lectures in diverse areas pertaining to software quality, advanced testing techniques and other areas of interest to software quality in public software conferences.

He is also associated as programmer in the development of Rapid Reporter tool.

Shmuel writes his thoughts around software testing at his own blog http://testing.gershon.info/ .

He can be contacted on Twitter @sgershon .

turning the tide of Bad Testing (Part 2)

by Martin Jansson

In my last article in **Tea-time with Testers – March Issue** I discussed about the things that increase the Testing Debt. Well, don't be discouraged, it is possible to fixing the broken windows and decreasing the testing debt.

What do we do to contribute? What do we do to provide value? Where do you start?

What can you do to decrease the Testing Debt?

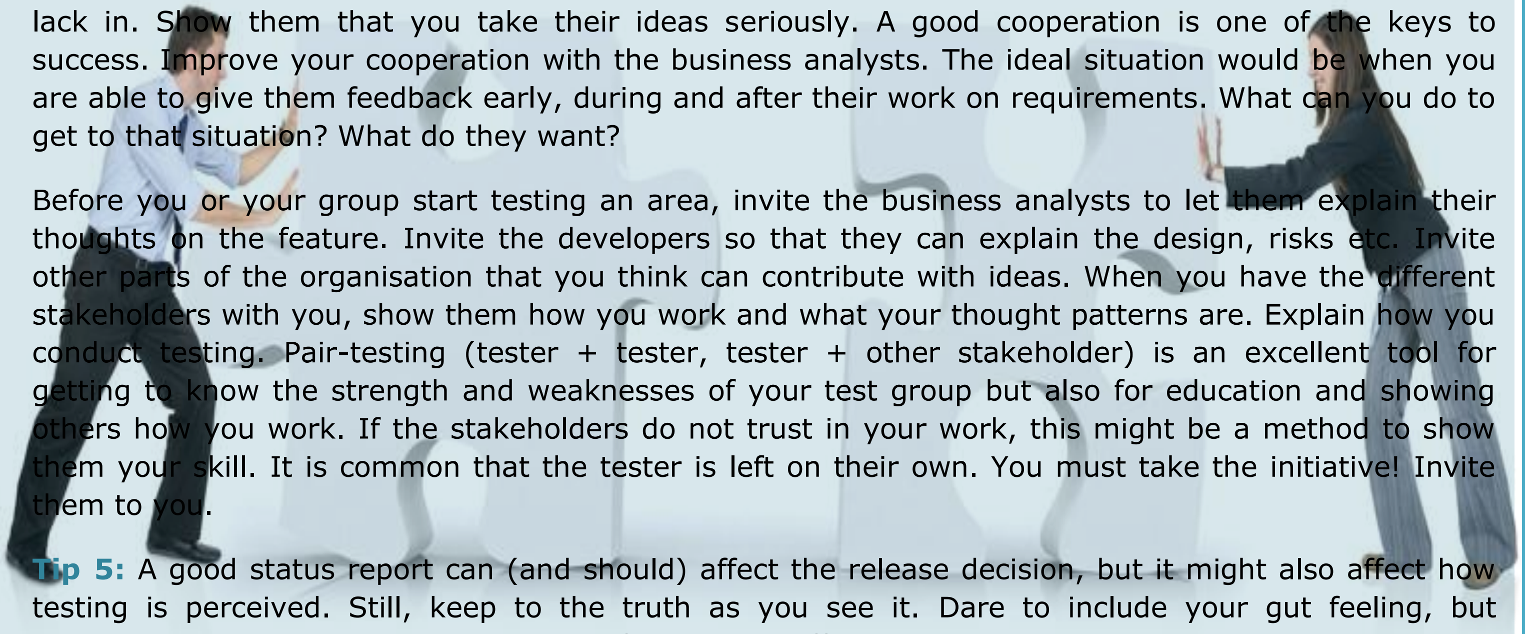
There are lots of things that you and your team can work on and excel in. There are also some areas which you can start with directly without depending on anyone but yourself and those you interact with.

Tip 1: Exploratory test perspective instead of script-based test perspective. This can be roughly summarized as more freedom to the testers, but at the same time adding accountability to what they do (See A tutorial in exploratory testing by Cem Kaner for an excellent comparison). And most importantly, the intelligence is NOT in the test script, but in the tester. The view on the tester affect so many things. For instance, where you work … can any tester be replaced by anyone in the organization? This means that your skill and experience as a tester is not really that important? If this is the case, you need to talk to those who support this view and show what you can do as a tester that makes his/her own decisions. Most organisations do not want robots or non-thinking testers who are affecting the release.

Tip 2: Focus on what adds value to developers, business analysts and other stakeholders. If you do not know what they find valuable, perhaps it is time that you found out!

Tip 3: Grow into a jelled team (read Peopleware by Timothy Lister and Tom deMarco for inspiration). Peopleware identifies, among other things, things that you should NOT do in order for a group to grow into a jelled team. As a team it is so much easier to gain momentum in testing and especially so if you are jelled. Do you need to reconsider how you work as a team?

Tip 4: Work on your cooperation. Improve your cooperation with the developers. Assist them in what they think is hard or not their job. Polish on the areas of improvement that the developer think you lack in. Show them that you take their ideas seriously. A good cooperation is one of the keys to success. Improve your cooperation with the business analysts. The ideal situation would be when you are able to give them feedback early, during and after their work on requirements. What can you do to get to that situation? What do they want?



Before you or your group start testing an area, invite the business analysts to let them explain their thoughts on the feature. Invite the developers so that they can explain the design, risks etc. Invite other parts of the organisation that you think can contribute with ideas. When you have the different stakeholders with you, show them how you work and what your thought patterns are. Explain how you conduct testing. Pair-testing (tester + tester, tester + other stakeholder) is an excellent tool for getting to know the strength and weaknesses of your test group but also for education and showing others how you work. If the stakeholders do not trust in your work, this might be a method to show them your skill. It is common that the tester is left on their own. You must take the initiative! Invite them to you.

Tip 5: A good status report can (and should) affect the release decision, but it might also affect how testing is perceived. Still, keep to the truth as you see it. Dare to include your gut feeling, but expressing it clearly that it is just that. If you have different metrics included, be sure to add context and how you as a tester interpret it.

Tip 6: The bug report is one of the most important artifacts that come from the tester. A bad bug report can have so much negative impact, while a good one can have the opposite. If possible, review the bugs before you send them. By doing this you will get new test ideas as well as raising the quality of the bug report. Train your skill in reporting bugs. Notify project management, developers, etc that NO bug report with bad quality is to be accepted from your team and that you want feedback on how to improve. You really want to make the life easier for the bug classification team and developers trying to pin point and eventually fix the bugs.

In a project a co-worker and I had worked on a bug for a time. It was a late Friday afternoon and almost everyone were heading home. It was only a few days before the release. Each bug reported engaged lots of people, no matter how small they were. The bug we found though was a blocker. We considered if we were going to hand in the bug as it was or if we were going to go to the roots of it. A week or two earlier the developers had worked the whole weekend trying to fix a set of bugs that were very badly written and hard to reproduce. So, we decided that we were going to collect as much information as we possibly could for the bug to get a good classification and possibly get fixed. Our goal was to make a good bug report. We started keeping track of the frequency and noticed that it was 10 out 50 times. We collected logs and reports from all parts of the system. When we knew that the repro-steps were correct, as we saw it, and that the content was ready. Then the bug was released into the bug system. After an analysis by the bug classification team the determined that the bug was both hardware and software related, so several teams got involved. When hardware thought they were ready they moved the bug to software. It was possible to track the communication and collaboration between the different teams. Not once were there any hesitation that any information was missing or that it was incorrect. In total there were close to 30 people working on the bug. Eventually it all got fixed and was sent back for verification. So, we spent a few extra hours to make a good bug report and saved/minimized time as well as lessening frustration.

Most of these tips are easy to start with, but they are also important to work with continuously.

Summary:

- Raise your ambition level
- Care about your work and those that you work with
- Prioritize testing before administration
- Cooperate and collaborate
- Report World class bugs
- Create status reports that adds value

DO NOT LIVE WITH BROKEN WINDOWS!

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 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 or on martin.jansson@qamcom.se

testing intelligence

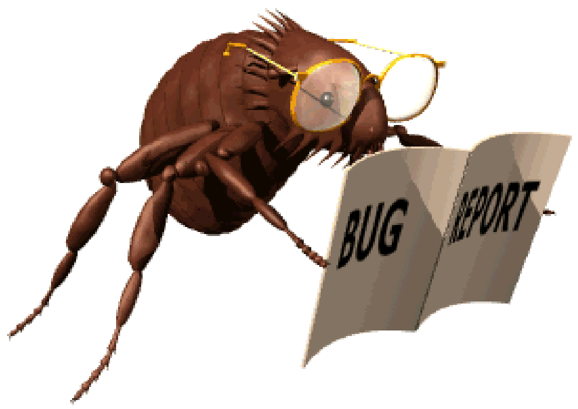
- its all about becoming an intelligent tester

an exclusive series by **Joel Montvelisky**

Principles of Good Bug Reporting…!

Once I heard a developer commenting on a couple of testers saying that he was annoyed by their low level of professionalism.

Since I actually know both of them I could understand his comment regarding one of them, but the second one is actually a great tester…



So I asked the developer and he told me that whenever he got a bug from the second tester he was always missing something. Sometimes it was vague steps to reproduce, other times it was attachments with captures of the error messages, and on top of that every bug this guy reported was ALWAYS urgent – never correctly prioritized.

Reporting a good bug is not hard, but many times we do it hurriedly and without putting our full attention into it.

As in the case above, the results of a badly written bug are 3-fold:

1. You waste your time reporting a bug incorrectly and most certainly will need to re-write it.
2. You waste the time of the other people reviewing the bug and trying to make sense of what you wrote.
3. You harm your good name as a QA Professional, by writing a bug that is either correct but incomprehensible or altogether wrong.

What are the components of a Good Bug?

1. Short and precise title

Each bug should have its individual title. The title gives us the opportunity to quickly understand and remember what the bug is all about.

The title cannot be generic (e.g. Error on Application), or too long. It should be one sentence, providing the highlights of the issue and the way to understand how a user could run into it.
Examples of good and bad titles are (taken from actual bugs!): "Unable to log into the system when username contains foreign characters" and not "User cannot work in Chinese"; or "Log rotation doesn't work, grows until the disc runs out of space" and not "Server crashes due to log size".

2. Concise but complete description: Steps to reproduce, consequences of the bug, and suggested behavior

The description of the bug is where the tester can express his "artistic sense" and write what he thinks will help stakeholders to understand and handle the bug.

The most important things to include in the bug are:

i. Steps to reproduce – with all the steps, sometimes even the trivial ones, just to make sure people who need to judge the bug and are not experts in the system can understand it correctly. Make sure to include also the data required to reproduce it (e.g account names, strings, etc).

ii. Consequences of the bug – what will happen to the user if/when she runs into it.
Here again, sometimes it is trivial but other times it isn't and you need to make sure this information reaches the person who is reading your report.

iii. Suggested/Expected behavior – especially when you are reporting a bug that doesn't create data-loss or other catastrophes, but on the other hand makes the application uncomfortable or unusable to the end-user.

The description is the place where you can add all the information that you cannot write on any other place, but on the other hand you cannot write to much into it (and expect people to read it) or write irrelevant information.

As a rule of thumb, 3 to 10 lines should be enough in 80% of the cases.

3. Good attachments

Especially true when you need to show GUI malfunctions, error messages, and/or log files that compliment the description of your bug.

One thing to take into account is to provide ONLY the relevant information in your attachment. This means that if you have a long log file attach only what you know is relevant and not all of it, if you have a screen capture cut down the area that is relevant and not all the desktop if it is unnecessary, etc .Use also zip files and compressed formats in order to save space.

4. Complete definition of the categorizing fields

If you work with a Structured Bug tracking system you will have fields that categorize your issues. Examples of these fields are Module, Infrastructure, Browser, etc.
These fields will help your team to categorize and in some cases reproduce the bug correctly. The only reasons not to fill these fields are laziness or carelessness.

5. Correct Severity & Priority

Another big *Tester Sin* is to over prioritize your bugs. Obviously you are happy you found a bug and you want it fixed, but giving it the wrong severity will only make the person in charge of the bug angry and it will lower your chances of having this bug fixed.

You will also automatically harm your name with this person and from now on she will be skeptic whenever you report a critical issue (remember "the boy who cried wolf"…)

If your problem is that you are not sure what severity to give your bug you can either consult with a fellow tester or developer, or ask you lead for a table describing the cases and standards used in your company for this purpose.

6. Follow-up and comment

You should continue following up on your bugs, and providing comments when necessary. This is especially true when a developer or other stakeholder does not understand the bug, and either rejects it or delays it.

You are not the owner of your bugs, but you certainly have say on them and you should make sure that say is heard. Depending on your company this last point may not apply, but I think in most companies it does.

To conclude, writing a bug is not Rocket Science but it requires you to concentrate on what you do and to make sure you don't abuse or become careless with what you are doing. Just put your head into it…

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 int he 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

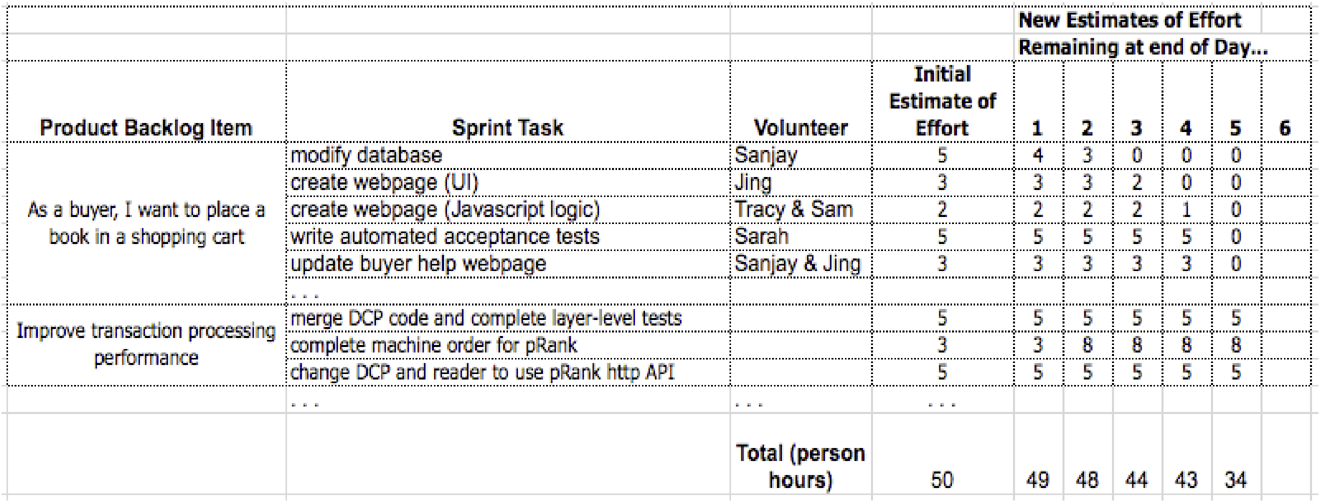
THE SCRUM PRIMER PART 3

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

Readers are encouraged to read the first 2 parts of this series in our Jan & Feb Issue – Editor

Updating Sprint Backlog & Sprint Burndown Chart

The Team in Scrum is self-managing, and in order to do this successfully, it must know how itis doing. Every day, the Team members update their estimate of the amount of time remaining to complete their current task in the Sprint Backlog (Figure 6). Following this update, someone adds up the hours remaining for the Team as a whole, and plots it on the Sprint Burndown Chart (Figure 7). This graph shows, each day, a new estimate of how much work (measured in person hours) remains until the Team's tasks are finished. Ideally, this is a downward sloping graph that is on a trajectory to reach "zero effort remaining" by the last day of the Sprint. Hence it is called a burndown chart. And while sometimes it looks good, often it does not; this is the reality of product development. The important thing is that it shows the Team their progress towards their goal, not in terms of how much time was spent in the past (an irrelevant fact in terms of progress), but in terms of how much work remains in the future –what separates the Team from their goal. If the burndown line is not tracking downwards towards completion near the end of the Sprint, then the Team needs to adjust, such as to reduce the scope of the work or to find a way to work more efficiently while still maintaining a sustainable pace. While the Sprint Burndown chart can be created and displayed using a spreadsheet, many Teams find it is more effective to show it on paper on a wall in their workspace, with updates in pen; this "low-tech/high-touch" solution is fast, simple, and often more visible than a computer chart.



				New Estimates of Effort					
				Remaining at end 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	Sanjay	5	4	3	0	0	0	
	create webpage (UI)	Jing	3	3	3	2	0	0	
	create webpage (Javascript logic)	Tracy & Sam	2	2	2	2	1	0	
	write automated acceptance tests	Sarah	5	5	5	5	5	0	
	update buyer help webpage	Sanjay & Jing	3	3	3	3	3	0	
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5	5	5	5	5	5	
	complete machine order for pRank		3	3	8	8	8	8	
	change DCP and reader to use pRank http API		5	5	5	5	5	5	
	. . .	. . .	. . .						
		Total (person hours)	50	49	48	44	43	34	

Figure 6. Daily Updates of Work Remaining on the Sprint Backlog

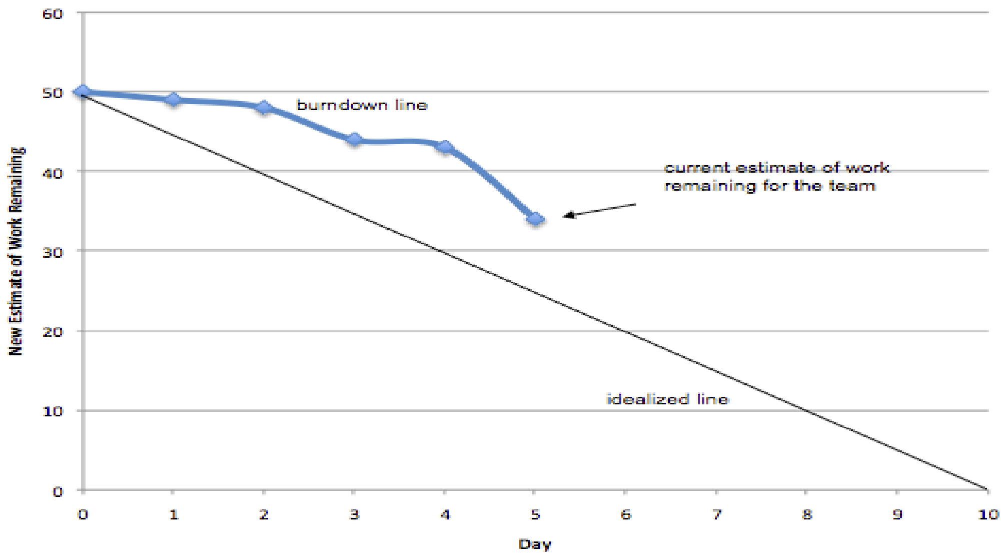


Figure 7. Sprint Burndown Chart

Product Backlog Refinement

One of the lesser known, but valuable, guidelines in Scrum is that five or ten percent of each Sprint must be dedicated by the Team to refining (or "grooming") the Product Backlog. This includes detailed requirements analysis, splitting large items into smaller ones, estimation of new items, and re-estimation of existing items. Scrum is silent on how this work is done, but a frequently used technique is a focused workshop near the end of the Sprint, so that the Team and Product Owner can dedicate themselves to this work without interruption. For a two week Sprint, five percent of the duration implies that each Sprint there is a half-day Product Backlog Refinement workshop. This refinement activity is not for items selected for the current Sprint; it is for items for the future, most likely in the next one or two Sprints. With this practice, Sprint Planning becomes relatively simple because the Product Owner and Scrum Team start the planning with a clear, well-analyzed and carefully estimated set of items. A sign that this refinement workshop is not being done (or not being done well) is that Sprint Planning involves significant questions, discovery, or confusion and feels incomplete; planning work then often spills over into the Sprint itself, which is typically not desirable.

Ending the Sprint

One of the core tenets of Scrum is that the duration of the Sprint is never extended – it ends on the assigned date regardless of whether the Team has completed the work it committed to. A Team typically over-commits in its first few Sprints and fails to accomplish its commitments. Sometimes it then overcompensates and under-commits, and finishes early (in which case it can ask the Product Owner for more Product Backlog items to work on). But by the third or fourth Sprint a Team has typically figured out what it is capable of delivering (most of the time), and they will meet their Sprint goals more reliably after that. Teams are encouraged to pick one duration for their Sprints (say, two weeks) and not change it. This helps the Team learn how much it can accomplish, which helps in both estimation and longer term release planning. It also helps the Team achieve a rhythm for their work; this is often referred to as the "heartbeat" of the Team in Scrum.

Sprint Review

After the Sprint ends, there is the Sprint Review, where the Team and the Product Owner review the Sprint. This is often mislabeled the "demo" but that does not capture the real intentof this meeting. A key idea in Scrum is inspect and adapt. To see and learn what is going on and then evolve based on feedback, in repeating cycles. The Sprint Review is an inspect and adapt activity for the product. It is a time for the Product Owner to learn what is going on with the product and with the Team (that is, a review of the Sprint); and for the Team to learn what is going on with the Product Owner and the market. Consequently, the most important element of the Review is an in-depth conversation between the Team and Product Owner to learn the situation, to get advice, and so forth. The review includes a demo of what the Team built during the Sprint, but if the focus of the review is a demo rather than conversation, there is an imbalance. A useful – but often overlooked – Scrum guideline is that it the ScrumMaster's responsibility to ensure that everyone knows the "Definition of Done" defined for this product or release. He prevents the team from demonstrating or discussing Product Backlog Items that are not 'done' according to the "Definition of Done." Items that are not 'done' go back to the Product Backlog and will be re-prioritized by the Product Owner. In way, there is transparency regarding the quality of the work; Teams cannot fake the quality by presenting software that appears to work well, but may be implemented with a messy pile of poor quality and untested code.

Present at this meeting are the Product Owner, Team members, and ScrumMaster, plus customers, stakeholders, experts, executives, and anyone else interested. The demo portion of the Sprint Review is not a "presentation" the Team gives – there is no slideware. A guideline in Scrum is that no more than 30 minutes should be spent preparing for the review, otherwise it suggests something is wrong with the work of the Team. It is simply a demo of what has been built. Anyone present is free to ask questions and give input.

Sprint Retrospective

The Sprint Review involves inspect and adapt regarding the product. The Sprint Retrospective, which follows the Review, involves inspect and adapt regarding the process. This is a practice that some Teams skip, and that's unfortunate, because it's the main mechanism for taking the visibility that Scrum provides into areas of potential improvement, and turning it into results. It's an opportunity for the Team to discuss what's working and what's not working, and agree on changes to try. The Team and ScrumMaster will attend, and the Product Owner is welcome but not required to attend. Sometimes the ScrumMaster can act as an effective facilitator for the retrospective, but it may be better to find a neutral outsider to facilitate the meeting; a good approach is for ScrumMasters to facilitate each others' retrospectives, which enables cross-pollination among Teams. There are many techniques for conducting a Sprint Retrospective, and the book Agile Retrospectives (Derby, Larsen 2006) provides a useful catalogue of techniques. A simple way to structure the discussion is to draw two columns on a whiteboard, labeled "What's Working Well" and "What Could Work Better" – and then go around the room, with each person adding one or more items to either list. As items are repeated, check marks are added next to them, so the common items become clear. Then the Team looks for underlying causes, and agrees on a small number of changes to try in the upcoming Sprint, along with a commitment to review the results at the next Sprint Retrospective. A useful practice at the end of the Retrospective is for the Team to label each of the items in each column with either a "C" if it is caused by Scrum (in other words, without Scrum it would not be happening), or an "E" if it is exposed by Scrum (in other words, it would be happening with or without Scrum, but Scrum makes it known to the Team), or a "U" if it's unrelated to Scrum (like the weather). The Team may find a lot of C's on the "What's Working Well" side of the board, and a lot of E's on the "What Could Work Better "; this is good news, even if the "What Could Work Better" list is a long one, because the first step to solving underlying issues is making them visible, and Scrum is a powerful catalyst for that.

Updating Release Backlog & Burndown Chart

At this point, some items have been finished, some have been added, some have new estimates, and some have been dropped from the release goal. The Product Owner is responsible for ensuring that these changes are reflecting in the Release Backlog (and more broadly, the Product Backlog). In addition, Scrum includes a Release Burndown chart that shows progress towards the release date. It is analogous to the Sprint Burndown chart, but is at the higher level of items (requirements) rather than fine-grained tasks. Since a new Product Owner is unlikely to know why or how to create this chart, this is another opportunity for a ScrumMaster to help the Product Owner. See Figure bellow for an example of the Release Backlog chart.

					New Estimates of Effort Remaining at end of Sprint...					
Item	**Details (wiki URL)**	**Priority**	**Estimate of Value**	**Initial Estimate of Effort**	**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	0	0	0			
As a buyer, I want to remove a book in a shopping cart	...	2	6	2	0	0	0			
Improve transaction processing performance (see target performance metrics on wiki)	...	3	6	13	13	0	0			
Investigate solutions for speeding up credit card validation (see target performance metrics on wiki)	...	4	6	20	20	20	0			
Upgrade all servers to Apache 2.2.3	...	5	5	13	13	13	13			
Diagnose and fix the order processing script errors (bugzilla ID 14823)	...	6	2	3	3	3	3			
As a shopper, I want to create and save a wish list	...	7	7	40	40	40	40			
As a shopper, I want to to add or delete items on my wish list	...	8	4	20	20	20	20			
. . .				. . .	. . .					
			Total	**537**	**580**	**570**	**500**			

(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

Gabrielle Benefield 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

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

Tickle your Testing Bone ! ☺

Question 1: How many testers does it take to change a light bulb?

Answer: None. Testers do not fix problems; they just find them.

Question 2: How many programmers does it take to change a light bulb?

Answer 1: What's the problem? The bulb at my desk works fine!

Answer 2: None. That's a hardware problem.

Call for Articles !

Have you got something to say?

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

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

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

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

sciencephotogallery

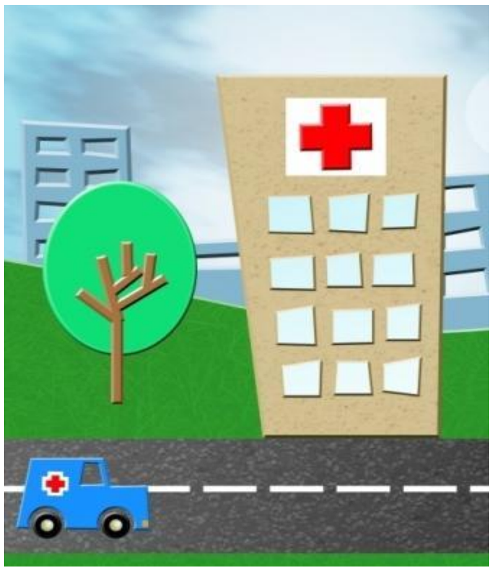
The Tale of Two Doctors

In a big city lived Joe, a typical urban yuppie. He was always focused on a great tomorrow. He worked very hard, partied furiously and lived a fast life.

Life was a blast, until his body decided to act up. On this Sunday morning, he woke up panting, unable to breathe, body drenched in sweat, with a dull pain in his chest. The previous evening was a blast, a celebration party thrown for his best buddy getting engaged. After an evening spent at bowling, they hit the pubs, closing each one, until they could not find one open.

A typical Sunday morning would commence at noon; today, as he was rudely jolted out of his reverie, the bright LED clock showed 7:00. He could not move his arms; it seemed to take a tremendous effort to reach out for the bottle of Evian on the table near his bed. He had read about old age diseases getting younger in these modern times, dismissed them brashly, a reflection of his supreme yuppie confidence. For once he faltered, worried seriously if he could become one of the stories. All these years, he had thought of God of as a fashion statement but today he genuinely wished to believe that God exists. For the next few seconds, which seemed like an eternity, his mind rapidly flew back over the past years on the constant abuse he had heaped on his body. For once he prayed dearly that he would do the right things, if he was excused this time. The clock glowed 7:02, and he realized it had been the longest two minutes of his life.

At 8:55 a.m he was in the reception of GoodLife hospital for a 9AM appointment with Dr. Robert Black Sr., a senior and very experienced cardiologist. He was soon ushered in, and was face to face with a severe yet friendly gentleman, a few orderly strands of golden hair on his shiny head with a piercing pair of eyes. "Mr. Joe, would you please tell me your problem?" said Dr. Black. Joe described in detail his travails upon his rude awakening. An old school doctor, he believed in detailed physical examination rather than the fancy modern equipment. "Lie down on the bed and relax Mr. Joe". He took his stethoscope, placed it on his chest and listened carefully. "Breathe in and out deeply now" said the doctor as he continued to hover his strength on various parts of chest. His sharp eyes showed no emotion, as he went about his job confidently. The young nurse standing next to the doctor was petite and beautiful. She dispassionately took out the sphygmomanometer, wrapped the elastic band around Joe's arm, pumped it up and watched the mercury bobbing up and down, while her other hand measured the pulse. After a minute, she looked at Dr. Black and said 140/120 and 93, in a husky voice.



"Mr. Joe, have you been feeling very tired at the end of day lately?"

"Yes" he said and added "It is the busy time of the year at work, a string of late nights."

"What kind of work do you do Mr. Joe?"

"I work in the software industry. We are in midst of building a cool application for mobile phones"

"Oh I see, you are the software guy. My nephew is in the software business too and is always racing against time." "I guess you must be tied to the desk most of the time. Do you exercise?"

"Well doctor, the days are long and busy, I catch up on my sleep over the weekend. I try to work out in the gym over the weekends, but it is challenging"

"I see that you are smoker, do you Drink? And are you a vegetarian?"

"Well doctor, I do Drink, and I am not vegetarian."

Dr. Black was one of the most famous cardiologists in the country. He was a master at diagnosis, believed in scientific and systematic study of symptoms and their connections. He placed his gold-rimmed glasses on the table, rubbed his finger on his chin, leaned back on his cozy leather chair and his piercing eyes looked straight into Joe and said "Mr. Joe, you have issue with the blood supply to the heart muscles, there seems a advanced arterial block. It is necessary that you undergo angioplasty, a procedure to relieve the constriction very quickly within a fortnight."

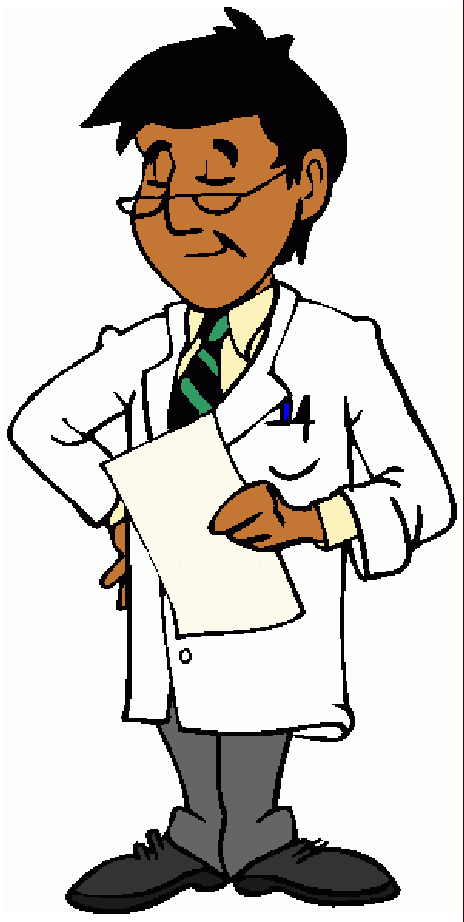
Joe sat still, staring at the statement "The most amazing non-stop machine. Take care." written on a poster containing the picture of heart, which prominently on the wall behind the doctor.

Dr. Black, who was used to these reactions, jolted Joe out of the reverie "Mr. Joe, you need to quit smoking and go vegetarian. You are young, the angioplasty procedure has a high success rate and you should be back on to active life quickly." Dr. Black believed in conveying news straight, and expected patients to face reality and act on the problem. "Mr. Joe, you would need to be in the hospital for 3-4 days, do decide on the date quickly. As I mentioned before, it is important that you act on this within a fortnight. It is painless procedure and should be fairly straightforward in your case. Mr. Joe, do you have any questions?"

"No doctor, I have none" replied Joe mechanically, his ability to think numbed by the turn of events. As he exited the consulting room, the receptionist, a cheerful and bubbly woman in twenties whispered "Have a good day Mr. Joe" with a beautiful smile, a genuine one. It had the effect, and for a moment he felt cheerful and returned the smile, a little weakly though.

As soon as he was outside the hospital, his hand went mechanically to his shirt pocket containing the cigarette packet. "Smoking kills" said the packet loudly and he threw the packet on the wayside.

David, his roommate had just woken up as Joe returned to his apartment. "Hi, had breakfast? Got some muffins and bagels, want some?" said David. "No thanks" mumbled Joe.



After a few minutes David understood the reason for the strange behavior of his best friend. "Come on man, let us get a second opinion right away". David was one who never lost his cool and his level headed thinking in tough situations was one that helped his friends many a times.

At 10:50 they were at ValleyTech hospital for consultation with Dr. James White. He had been referred by David's boss, who had undergone a heart bypass a few months at ValleyTech.

At 11:05 Joe was called in. "Good morning Joe. Please sit down. I have read your case sheet and have a few questions. Do you have any recent ECGs?

"No doctor" said Joe.

"I know your company has a yearly health check plan for all, as our hospital administers it. So have you not taken it this year?"

"No doctor, the last few months has been very hectic and I have not had my yearly checkup yet" said Joe.

Dr. White was a modern doctor who relied on technology in diagnosis and treatment. A young cardiologist, he believed in seeing the 'internals' before the scalpel touched the body. He was amazed at the advancements in radiology and made it a point to recommend a few pictures to be taken before he touched a patient.

Unlike Dr. Black who believed in the power of external examination, Dr. White believed in the looking at internals for diagnosis and treatment. Dr. White had immense faith in scans, lab tests and preferred analyzing reports to spending time on and performing examinations on patients.

"Please get the ECG done now. I would like to see the report first. Thank you."

Joe went to the diagnostic lab in the adjoining building. When his turn came, he went inside, removed his shirt, the technician smeared jelly on his chest and proceeded to stick colorful leads at various points. In a few seconds, the needle was dancing, drawing patterns on the strip of paper. The technician looked at the squiggles on the paper with a bored expression, and after a few minutes, decided the machine has had its share of fun and switched it off. He tore the roll of paper, scanned it intently, and then proceeded to fold it and inserted into a cover. Joe was curious to know what the squiggles meant and asked "Is it normal?"

"It seems fine, expect for a small spike here" replied the technician. He had seen hundreds of such squiggles and knew exactly as to what was normal, but he was no doctor to interpret any abnormalities.

Joe stared at the report, the squiggles held a secret that Joe was scared about. "Hey, let's go meet to the doc" now said David breaking Joe's train of negative thoughts.

"Show me the report Joe" said Dr. White.

Dr. White held the strip of paper and rapidly scrolled it forward and then backwards.

"Were you treated for any heart related issue when you were young?" asked Dr. White.
"No" said Joe, scared to ask questions.

"Joe, there is a slight aberration in the ECG, it may be nothing to be worry about. To confirm this, I recommend that you get the 128-slice beating heart scan. This most advanced technology for diagnosis of heart related ailments that is available in the world and we are the only hospital in this city to have this. This gives a clear picture of the beating heart and enables clear diagnosis. And also take a chest X-ray too. I will be available until 1:00 PM, get it done right away and then see me". He wrote down the lab request and handed it over to Joe.

"Hello, I need you to get '128-slice beating heart scan' done. How much does it cost? Joe asked the grim looking gentlemen on the cash counter.

He was shocked at the cost of the hi-tech scan; it seemed to have enough digits to max his credit card. Joe looked at David conveying in his look "They are milking us".

Joe realized that the second opinion was going to be expensive and needed to think about this before he went on a diagnostic spree.

Joe was a professional software tester who diagnosed software for defects. He decided to chat with David over a cup of coffee to decide whether to go ahead with the expensive scan. David went to get the coffee while Joe sat down at the corner table, gazing at the birds fluttering over the little pond outside.

During this mindless gaze, staring at the chirpy birds, it suddenly flashed on him, the parallels between his profession and the doctor's. In his job, he used black box techniques that required him to examine the system externally and find defects. He relied on deep understanding of the intended behavior, observation of behavior ("symptoms") to design & refine his test cases. He had at times looked at internal information like architecture, technology, code structure to design test cases that were adept at catching issues related to structure.

His colleagues always used terms like "Black box testing" and "White box testing" and associated these two with system and unit levels respectively.

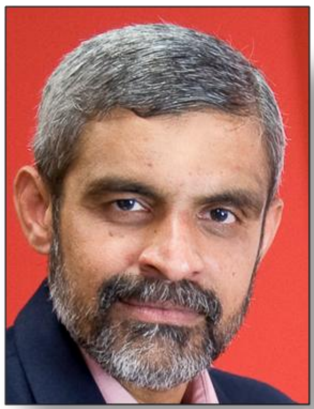
Now he realized the general misconception that unit testing was white box testing and system box testing was black box testing. His train of thought was interrupted by hot coffee spilling on his shoulder followed by the shrill sound of glass breaking. "I am really sorry, hope you are ok" said the elderly woman who had tripped over the protruding leg of the gentleman at the neighboring table and spilt the hot cup of expresso that she was carrying. "I am fine, let me help you" said Joe as he helped pick up the glass shreds.

He realized that certain types of defects were better caught via "internal examination"(white box test techniques) while some are most suited to be caught via "external examination" (black box test techniques). He now understood that both of these test techniques were required at all levels to uncover effectively and efficiently.

Suddenly the diagnosis approach followed by Dr. Robert Black and Dr. James White became clear. As soon as David laid the steaming cup of Cafe Latte on the table, Joe had made the decision. He was not withdrawn or worried. The confidence was back and he would not let the ECG scroll spoil his fun.

Note: Drop me a note at ash@stagsoftware.com or tweet me @ash_thiru if you liked this. Thank you.

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.

stag

finally the wait is over…

Announcing…!

Vivek Bisen as a **"Smart Tester of The Month"**

Hi Lalitkumar,

Thanks for appreciation! I have been working in the IT industry for last 5 years. Dedicated software test engineer; working in automation testing with tools like QTP, TDE, eTester, TestComplete etc. I like programming in VBScript and JavaScript. Have keen interest in software test processes, test planning, developing automation framework. Have worked for life-sciences projects. Currently working with Oracle Financial Solutions Ltd as an Associate Consultant in retail banking.

- Vivek Bisen (Mumbai, India)

"Blogger of The Month" award goes to Neha Narang

Hello All ,

I am working as software engineer in TCS. After completing my college from Maharana Pratap college of Technology , I joined TCS with loads of dreams and enthusiasm and zeal to perform . Did my schooling from Carmel Convent Sr. secondary school . In my life i have won many awards that includes Green India Quiz and various prizes in IIT - Roorkee .

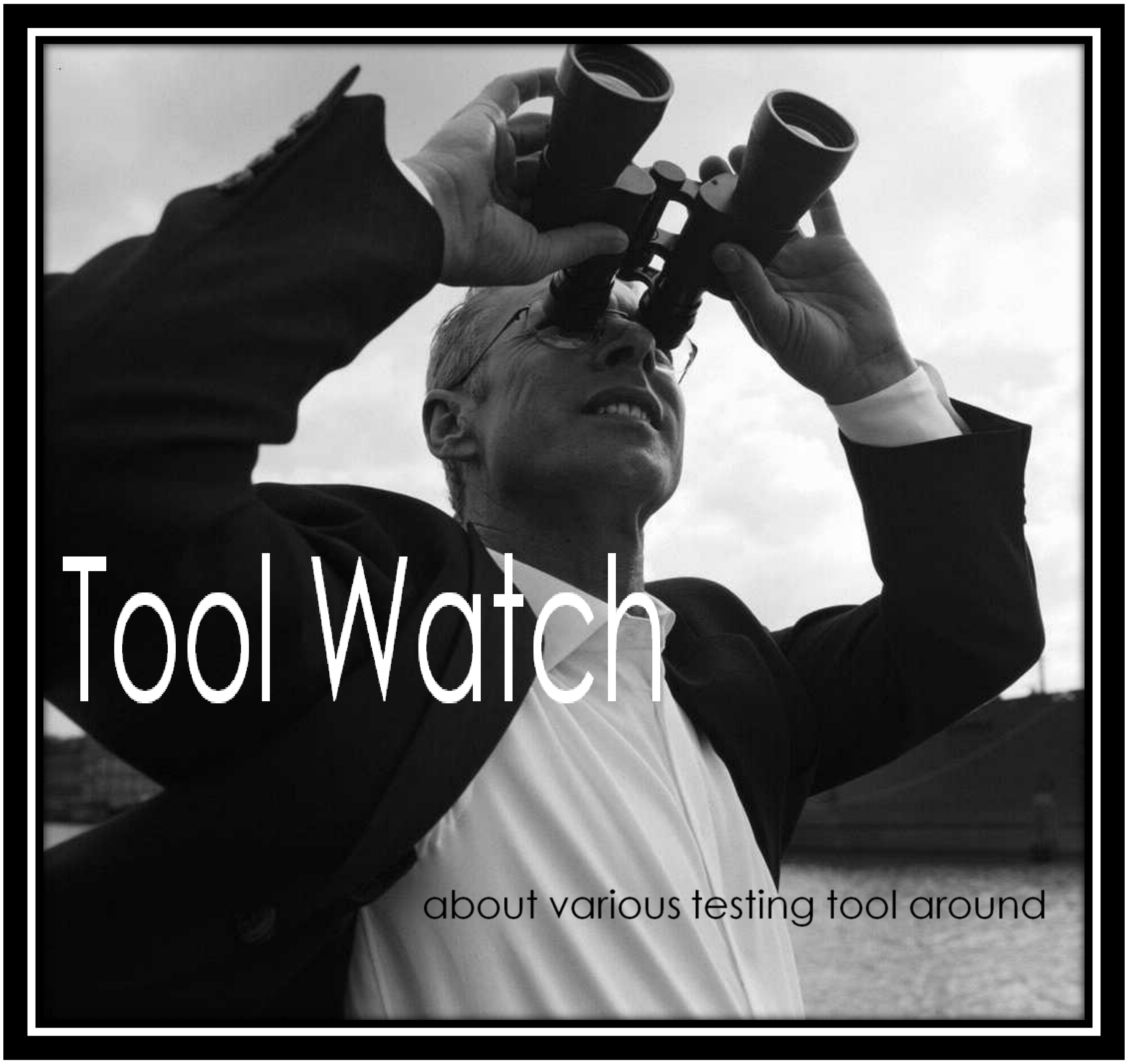
Though am a programmer by profession I have always loved the Tester in me and hence this blog post.

Thank you ***Tea-time with Testers*** for this wonderful initiative and thought provoking articles too.

Check out Neha's winning blog entry here

- Neha Narang (Bangalore, India)

Bug-Boss Challenge – Click here to know more


Tool Watch
about various testing tool around

PractiTest

Tea-time with Testers Rating: ★★★★☆

by Juhi Verma & Sharmistha Priyadarshini

Apart from the Requirement gathering, writing Test Cases & raising Bugs, we Testers also perform one equally important job i.e. Test Execution & Management in Test Management Tool.

No doubt, we all learn & adopt ourselves best to work with different Test Management Tools. But have we ever thought of something much simpler, interesting, innovative and definitely astonishing?

Well…won't it be an easy job just to have a look on Auto Generated Graphs of your Test Execution Status, Bug Statistics, Project Assignments to specific group and much more, that too just with a single sign on?

- What if your tool itself tells you that the bug you we are about to write is a duplicate of an existing bug?
- What if you can set the visibility of your project to other users?
- What if your tool is intelligent enough to handle the parent-child hierarchy of issues ?
- What if your tool provides the flexibility to the views according to the user using it?

And…How about managing your Tests and Bugs from your very own I-PAD?

Well, PractiTest is the Tool. Let us walk you through this best Test Management Tool we have ever come across.

Here you go.

1. About PractiTest

PractiTest provides you an end-to-end solution to cover your complete QA process. One can get all the information needed in one QA Management Platform.

An integrated QA Management System to help you manage your work and your team.

Software as a service (or SaaS or On Demand Software) is a way of delivering applications over the Internet—as a service. PractiTest enables you to access your project from anywhere via a browser, freeing yourself from worrying about the purchase, maintenance and updates of your software or hardware.

❖ Benefits of SaaS

PractiTest's SaaS benefits are many; here's a partial list:

- No Maintenance No Installations
- No Risk
- Affordable
- Better Access from Anywhere

❖ Security and Reliability

Our platform was designed to make sure only you and your users can access your information.

- **SSL Encrypted Connection**
  Access your projects using a secured connection, and be sure that no one can tap into your information while being transferred over the web.

- **Security Centric Architecture and Design**
  PractiTest is designed and build with all the security considerations in mind, and so only your users will be able to access your data. With the groups and permissions functionality you can also control who has access to what parts of the system and to do what operations.

- **Enterprise Level Hosting**
  We partner with some of the most serious vendors in the world to ensure our system will be up and always accessible from everywhere in the world.

- **Regular backups**
  PractiTest is backed up multiple times a day with the information being stored on geographically separated locations for better security.

2. Working with PractiTest

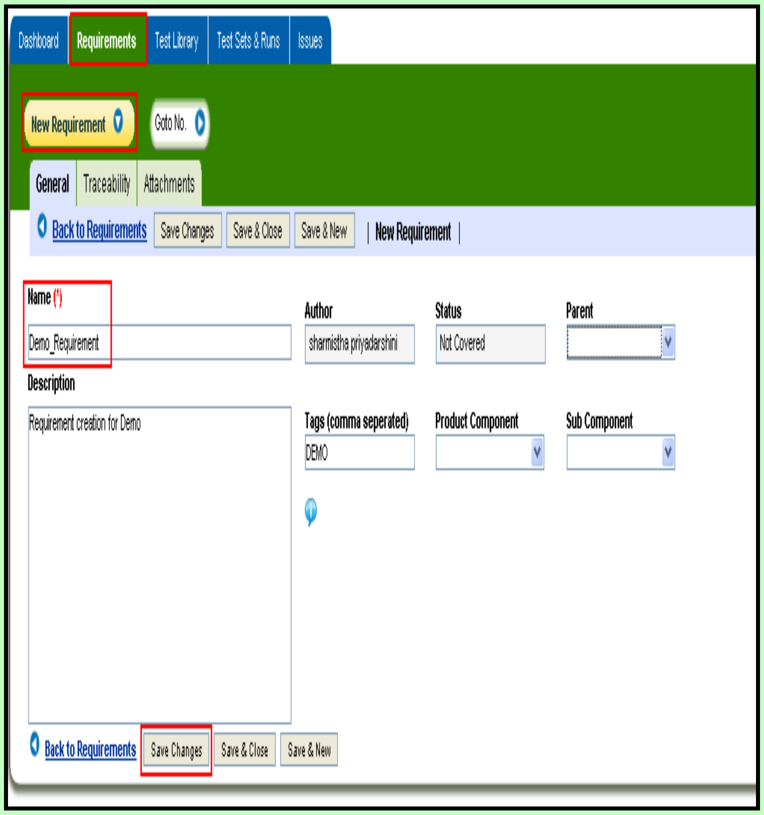
1) Creating New Requirement

The QA process starts by gathering the requirements for the System Under Test (SUT). You can capture requirements in PractiTest's Requirements module.

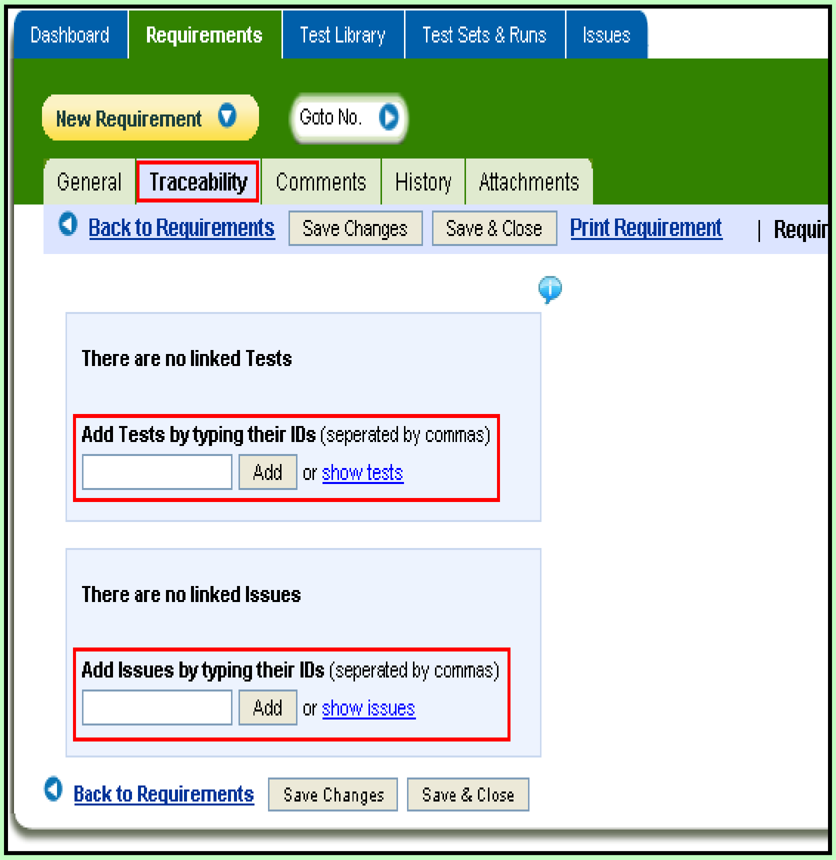
1. Go to the requirements tab.
2. Click on the new requirement button.
3. Enter the required details.
4. Click on Save Changes.

A new requirement is created with unique Id.

Notes:
A requirement can be linked to another requirement by providing the parent requirement name in the "Parent" text box.
Tag is used for advance search.



Name (*)	Demo_Requirement
Author	sharmistha priyadarshini
Status	Not Covered
Parent	
Description	Requirement creation for Demo
Tags (comma seperated)	DEMO
Product Component	
Sub Component	



There are no linked Tests

Add Tests by typing their IDs (seperated by commas)

There are no linked Issues

Add Issues by typing their IDs (seperated by commas)

Requirements Traceability:

Open the Requirement.

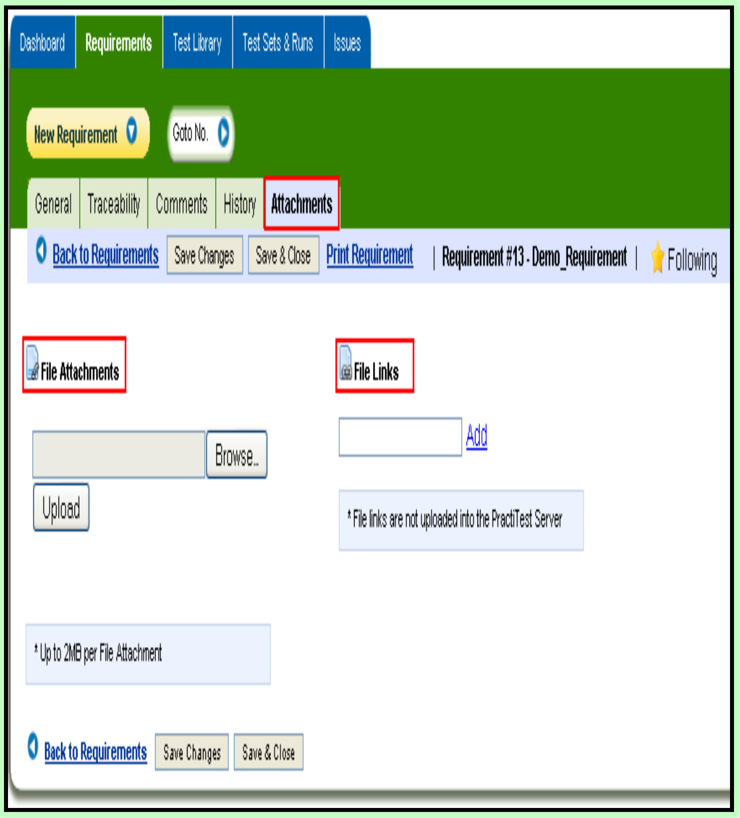
Click on Traceability Tab.

Here we can add test sets or issues to the requirements by clicking the ADD Button.

Also you can search for specific test/issue by clicking on Show Tests Link.

Requirements Attachments:

- We can upload files/documents.
- We can add live links/hyperlinks to the attachments

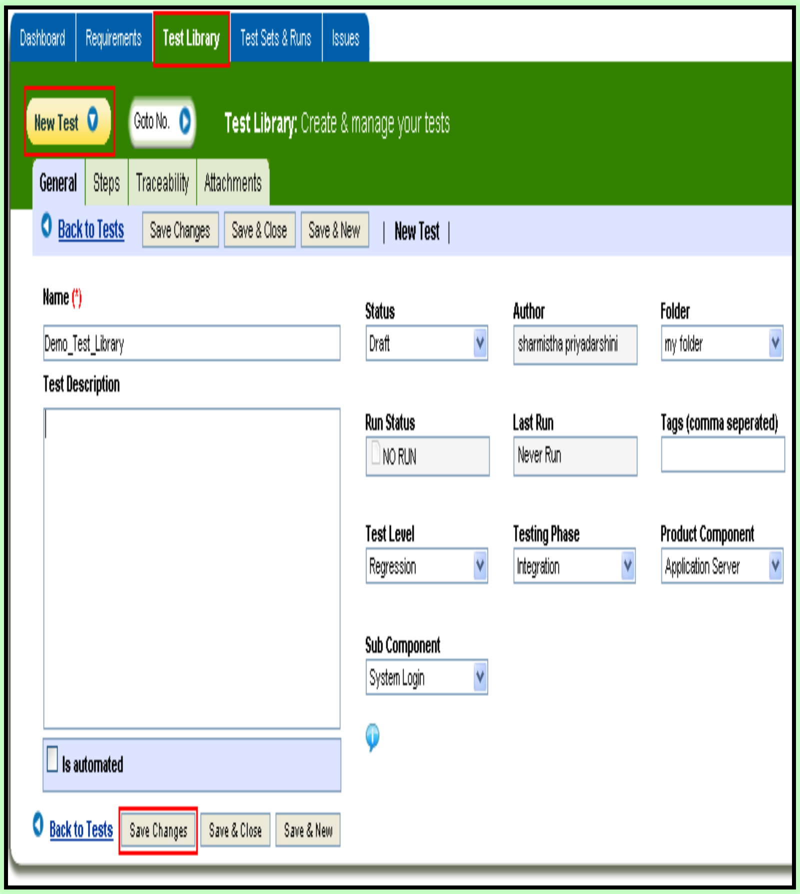


Requirement #13 - Demo_Requirement

* File links are not uploaded into the PractiTest Server

* Up to 2MB per File Attachment

2) Creating New Test Library:



Test Library: Create & manage your tests

New Test

Name (*)	Status	Author	Folder
Demo_Test_Library	Draft	sharmistha priyadarshini	my folder
Test Description	Run Status	Last Run	Tags (comma seperated)
	NO RUN	Never Run	
	Test Level	Testing Phase	Product Component
	Regression	Integration	Application Server
	Sub Component		
	System Login		

☐ Is automated

1. Go to the Test Library tab
2. Click on New Test button.
3. Enter required details.
4. Click on Save Changes button.

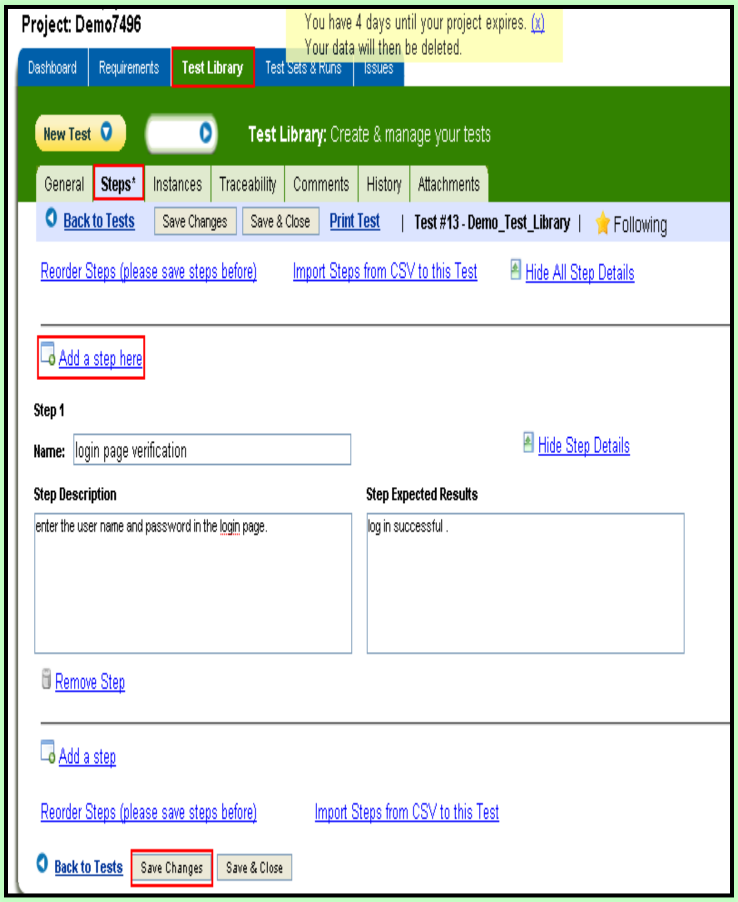
A new Test Library is created with unique Id.

Add Steps to Test Library:

1. Open the Test Library.
2. Select your Test Library from All Tests.
3. Click on Steps Tab.
4. Add Steps to the Test library by clicking the Add a Step Here button.
5. Save the changes.

Notes:

We can Add the test steps by uploading the file in .csv format.



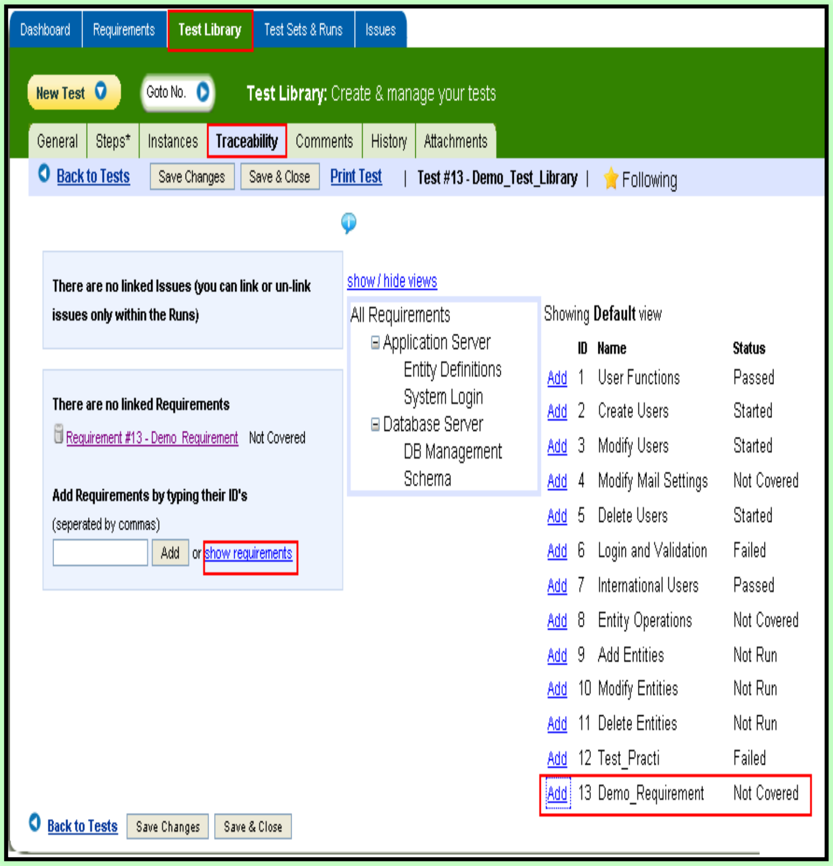
You have 4 days until your project expires. (x)
Your data will then be deleted.

Test #13 - Demo_Test_Library

Step 1

Name: login page verification

Step Description	Step Expected Results
enter the user name and password in the login page.	log in successful .



Test #13 - Demo_Test_Library

There are no linked Issues (you can link or un-link issues only within the Runs)

There are no linked Requirements

Requirement #13 - Demo_Requirement Not Covered

Add Requirements by typing their ID's (seperated by commas)

All Requirements
- Application Server
  - Entity Definitions
  - System Login
- Database Server
  - DB Management
  - Schema

Showing **Default** view

	ID	Name	Status
Add	1	User Functions	Passed
Add	2	Create Users	Started
Add	3	Modify Users	Started
Add	4	Modify Mail Settings	Not Covered
Add	5	Delete Users	Started
Add	6	Login and Validation	Failed
Add	7	International Users	Passed
Add	8	Entity Operations	Not Covered
Add	9	Add Entities	Not Run
Add	10	Modify Entities	Not Run
Add	11	Delete Entities	Not Run
Add	12	Test_Practi	Failed
Add	13	Demo_Requirement	Not Covered

Test Library Traceability:

1. Select the Test Library.
2. Select your Test Library from All Tests.
3. Click on Traceability Tab.
4. We can attach the requirements to the test library by typing their ID's or by clicking on show requirements link.
5. Save the Changes.

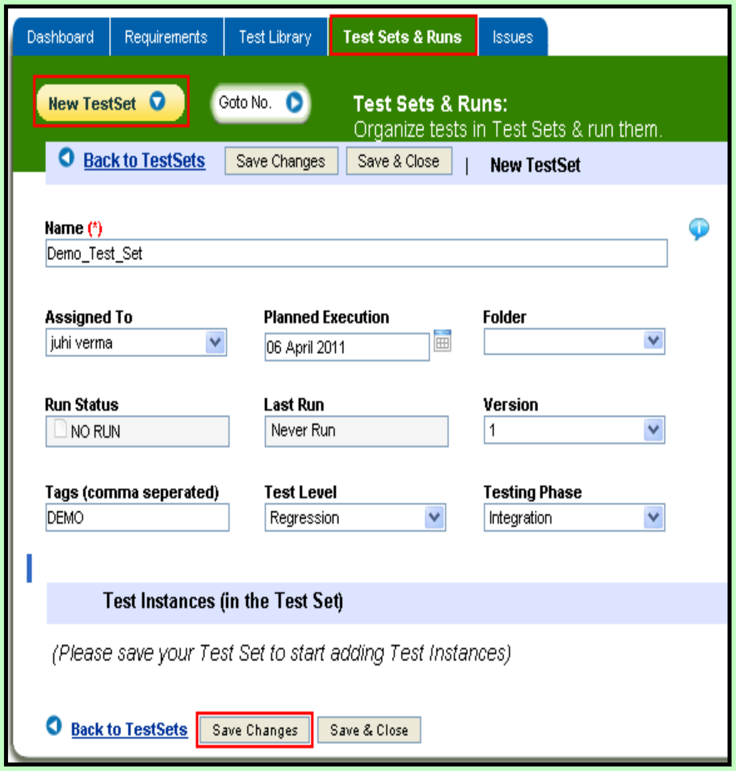
3) Creating a new Test Sets:

1. Go to the Test Sets & Runs tab
2. Select Test Sets&Runs Tab.
3. Click on New TestSet button.
4. Enter the required details.
5. Save the Changes.

Notes:

- One Test Set can contain multiple Tests.

One Test can be link to multiple Test Sets.



Test Sets & Runs:
Organize tests in Test Sets & run them.

New TestSet

Name (*)
Demo_Test_Set

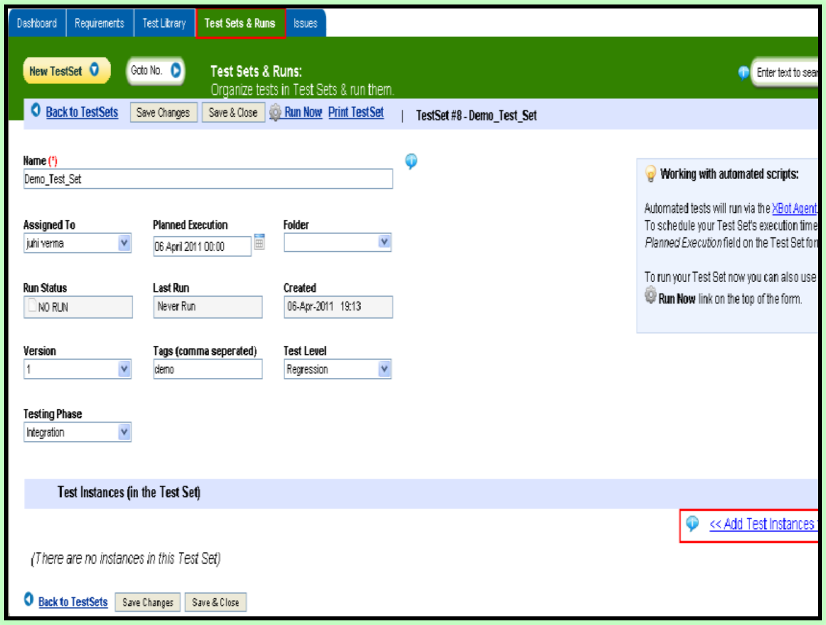
Assigned To: juhi verma
Planned Execution: 06 April 2011
Folder:

Run Status: NO RUN
Last Run: Never Run
Version: 1

Tags (comma seperated): DEMO
Test Level: Regression
Testing Phase: Integration

Test Instances (in the Test Set)

(Please save your Test Set to start adding Test Instances)



Test Sets & Runs:
Organize tests in Test Sets & run them.

TestSet #8 - Demo_Test_Set

Name (*)
Demo_Test_Set

Assigned To: juhi verma
Planned Execution: 06 April 2011 00:00
Folder:

Run Status: NO RUN
Last Run: Never Run
Created: 06-Apr-2011 19:13

Version: 1
Tags (comma seperated): demo
Test Level: Regression

Testing Phase: Integration

Working with automated scripts:

Automated tests will run via the XBot Agent
To schedule your Test Set's execution time
Planned Execution field on the Test Set for

To run your Test Set now you can also use
Run Now link on the top of the form.

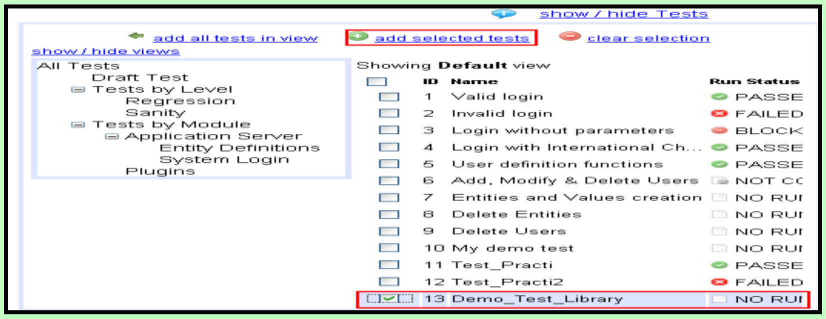
Test Instances (in the Test Set)

(There are no instances in this Test Set)

Add Test Instances to Test Set:

1. Select Test Sets & Runs tab.
2. Select your TestSet from All TestSets.
3. Click on Add Test Instances to Test Set.
4. Select test Library
5. Click on Add Selected tests.

Save the Changes.



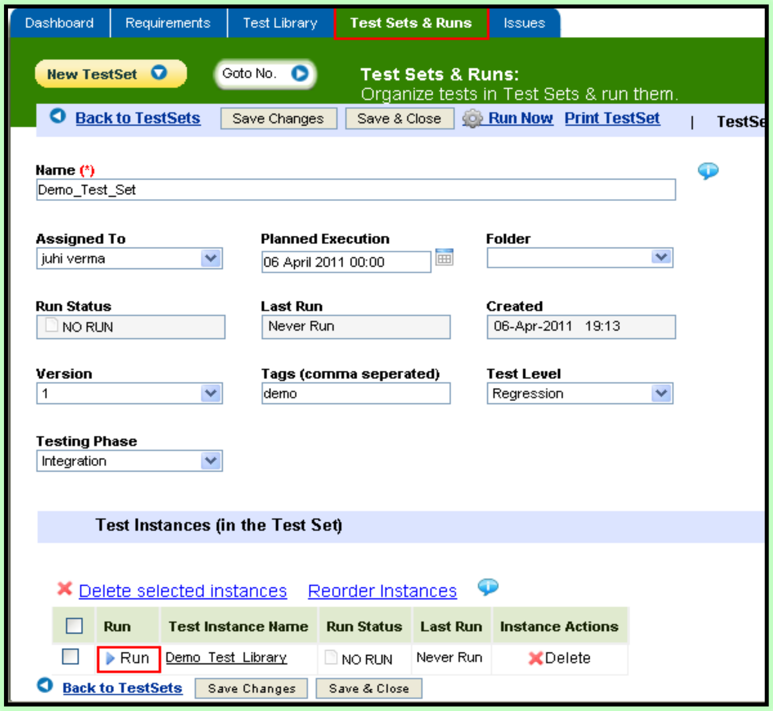
Showing **Default** view

ID	Name	Run Status
1	Valid login	PASSE
2	Invalid login	FAILED
3	Login without parameters	BLOCK
4	Login with International Ch...	PASSE
5	User definition functions	PASSE
6	Add, Modify & Delete Users	NOT CC
7	Entities and Values creation	NO RUI
8	Delete Entities	NO RUI
9	Delete Users	NO RUI
10	My demo test	NO RUI
11	Test_Practi	PASSE
12	Test_Practi2	FAILED
13	Demo_Test_Library	NO RUI

Run the Test sets:

1. Select Test Sets & Runs tab.
2. Select your TestSet from All TestSets.
3. Click on Run.

Save the Changes.



Name (*)
Demo_Test_Set

Assigned To	Planned Execution	Folder
juhi verma	06 April 2011 00:00	

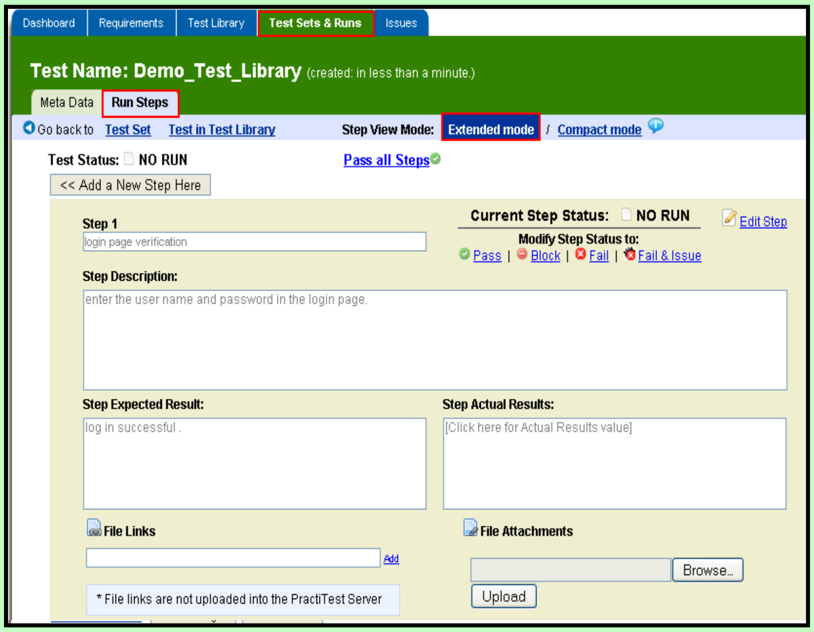
Run Status	Last Run	Created
NO RUN	Never Run	06-Apr-2011 19:13

Version	Tags (comma seperated)	Test Level
1	demo	Regression

Testing Phase
Integration

Test Instances (in the Test Set)

	Run	Test Instance Name	Run Status	Last Run	Instance Actions
☐	Run	Demo_Test_Library	NO RUN	Never Run	Delete



Test Name: Demo_Test_Library (created: in less than a minute.)

Test Status: NO RUN

Step 1
login page verification

Current Step Status: NO RUN

Modify Step Status to:

Step Description:
enter the user name and password in the login page.

Step Expected Result:
log in successful .

Step Actual Results:
[Click here for Actual Results value]

* File links are not uploaded into the PractiTest Server

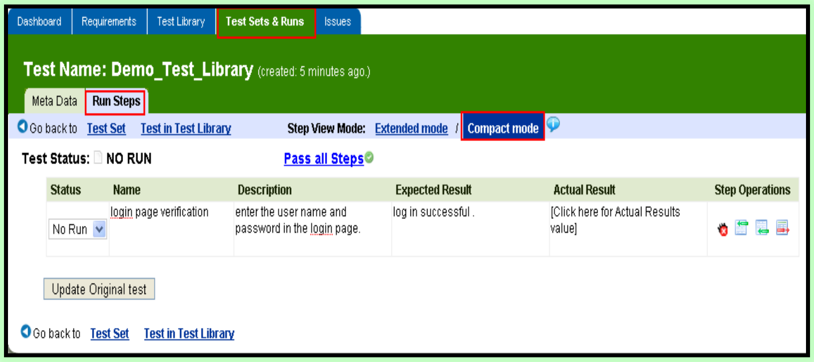
Run Steps for Test Sets:

1. Select Test Sets & Runs tab.
2. Select your TestSet from All TestSets.
3. Click on Run Steps Tab.

Here you have two options.

You can select any mode from Step View Mode:

Extended Mode and Compact Mode.



Test Name: Demo_Test_Library (created: 5 minutes ago.)

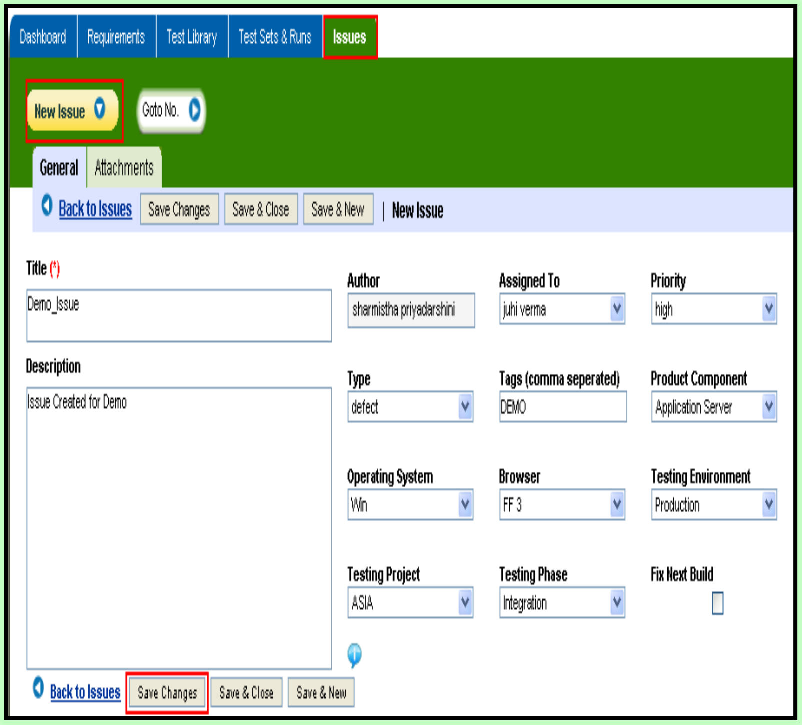
Test Status: NO RUN

Status	Name	Description	Expected Result	Actual Result	Step Operations
No Run	login page verification	enter the user name and password in the login page.	log in successful .	[Click here for Actual Results value]	

4) Creating a New Issue:

1. Select the Issues Tab.
2. Click on New Issue Button.
3. Enter required details.

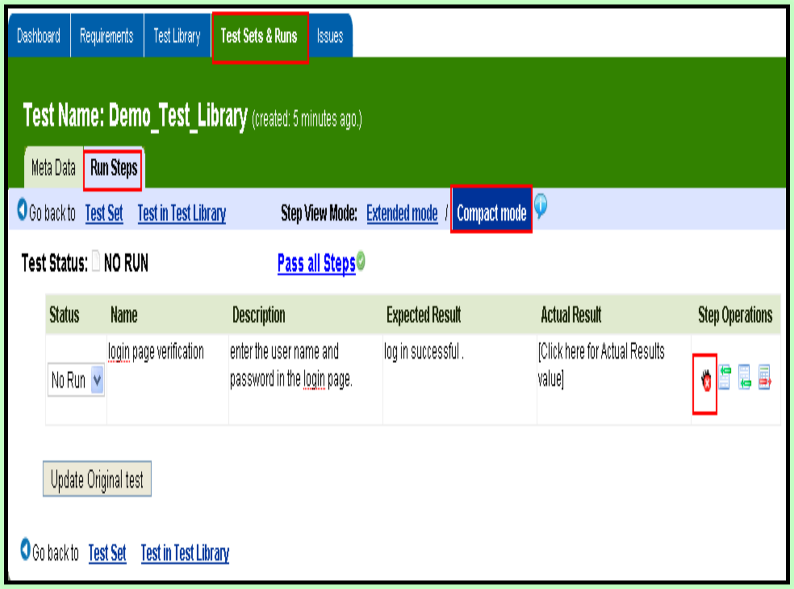
Save the changes.



Title (*): Demo_Issue

Description: Issue Created for Demo

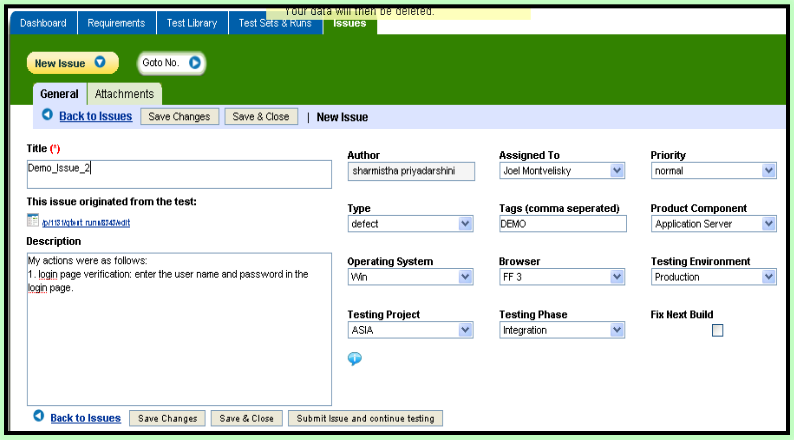
Author: sharmistha priyadarshini	Assigned To: juhi verma	Priority: high
Type: defect	Tags (comma seperated): DEMO	Product Component: Application Server
Operating System: Win	Browser: FF 3	Testing Environment: Production
Testing Project: ASIA	Testing Phase: Integration	Fix Next Build: ☐


Dashboard
Requirements
Test Library
Test Sets & Runs
Issues
Meta Data
Run Steps
Go back to Test Set Test in Test Library
Step View Mode: Extended mode / Compact mode
Pass all Steps
Update Original test
Go back to Test Set Test in Test Library

Test Name: Demo_Test_Library (created: 5 minutes ago.)

Test Status: NO RUN

Status	Name	Description	Expected Result	Actual Result	Step Operations
No Run	login page verification	enter the user name and password in the login page.	log in successful .	[Click here for Actual Results value]	



Title (*): Demo_Issue_2

This issue originated from the test: /p/1131/qtest_run/8343/edit

Description:
My actions were as follows:
1. login page verification: enter the user name and password in the login page.

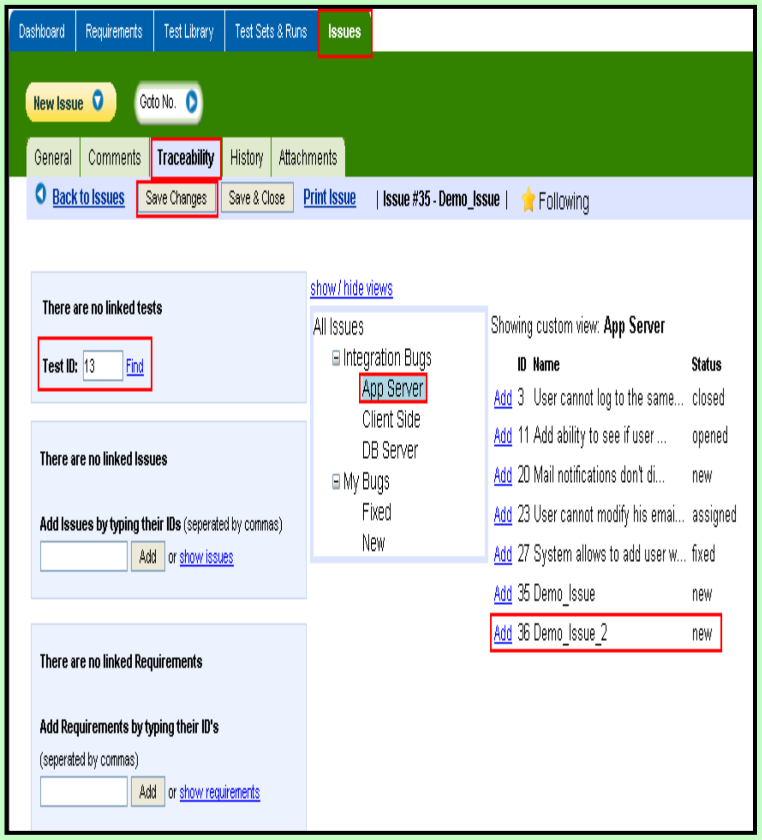
Author: sharmistha priyadarshini	Assigned To: Joel Montvelisky	Priority: normal
Type: defect	Tags (comma seperated): DEMO	Product Component: Application Server
Operating System: Win	Browser: FF 3	Testing Environment: Production
Testing Project: ASIA	Testing Phase: Integration	Fix Next Build: ☐

Test Run in Compact mode:

1. Select Test Sets & Runs tab.
2. Select your TestSet from All TestSets.
3. Click on Run Steps Tab.
4. Click on Compact mode.
5. Now, In step Operations, Click on fail & Issue button. It will allow you to raise an issue.
6. Enter the required details of the issue. It will automatically attach this issue to the Test.
7. Save the Changes.

Issues Traceability:

1. Select Issues Tab.
2. Select your Issue from list of Issues.
3. Click on Traceability Tab.
4. Enter the Test ID, also you can find Test ID from Find link available.
5. Enter Issue ID or click on Show issues link to attach the issue.
6. In Show/hide views, you can select the Issue by particular category.
7. Add the Issue.



There are no linked tests

Test ID: 13 Find

There are no linked Issues

Add Issues by typing their IDs (seperated by commas)

Add or show issues

There are no linked Requirements

Add Requirements by typing their ID's (seperated by commas)

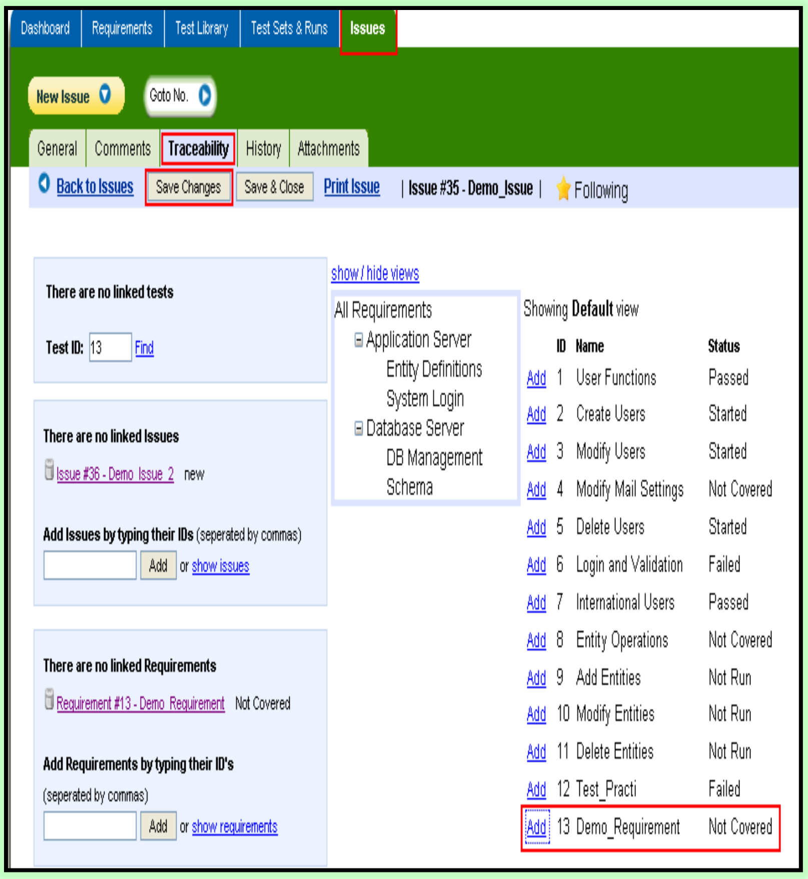
Add or show requirements

show / hide views

All Issues
- Integration Bugs
  - App Server
  - Client Side
  - DB Server
- My Bugs
  - Fixed
  - New

Showing custom view: **App Server**

	ID	Name	Status
Add	3	User cannot log to the same...	closed
Add	11	Add ability to see if user ...	opened
Add	20	Mail notifications don't di...	new
Add	23	User cannot modify his emai...	assigned
Add	27	System allows to add user w...	fixed
Add	35	Demo_Issue	new
Add	36	Demo_Issue_2	new



There are no linked tests

Test ID: 13 Find

There are no linked Issues

Issue #36 - Demo_Issue_2 new

Add Issues by typing their IDs (seperated by commas)

Add or show issues

There are no linked Requirements

Requirement #13 - Demo_Requirement Not Covered

Add Requirements by typing their ID's (seperated by commas)

Add or show requirements

show / hide views

All Requirements
- Application Server
  - Entity Definitions
  - System Login
- Database Server
  - DB Management
  - Schema

Showing **Default** view

	ID	Name	Status
Add	1	User Functions	Passed
Add	2	Create Users	Started
Add	3	Modify Users	Started
Add	4	Modify Mail Settings	Not Covered
Add	5	Delete Users	Started
Add	6	Login and Validation	Failed
Add	7	International Users	Passed
Add	8	Entity Operations	Not Covered
Add	9	Add Entities	Not Run
Add	10	Modify Entities	Not Run
Add	11	Delete Entities	Not Run
Add	12	Test_Practi	Failed
Add	13	Demo_Requirement	Not Covered

Issues Traceability:

1. In the same way we attach/link the requirements to the Issue.
2. Save the Changes.

To be continued in Next Issue- Editor

Biography

Juhi Verma is an Electronics Engineer working with Tata Consultancy Services (Mumbai) as a Tester. During her spell she has also worked as a programmer but she now prefers software Testing over programming as it's a dual fun, she says.

Testing is her passion and she enjoys participating and conducting testing related activities.

Juhi also works as a Team member at Tea-time with Testers.

She can be contacted at her personal mail id juhi_verma1@yahoo.co.in or on Twitter @Juhi_Verma .

Biography

Sharmistha Priyadarshini is currently working as a Test Engineer at Tata Consultancy Services (Mumbai).

Sharmistha is die hard lover of Data Base testing as she finds is challenging. Being a programmer in past she now loves software testing too as it gives her more scope for analysis.

Sharmistha can be reached via her mail id sharmistha.priyadarshini@tcs.com

Our Testimonials

Hi Team,

First of all heartiest congratulation for this wonderful initiative. Tea-time with Testers is the Best really best magazine I have ever read on software Testing. Never knew that a magazine on software testing can be this much astonishing and tempting to read.

Hard to believe that it was just second issue. It sounds like you are doing this fantastic job from ages.

Three Cheers to you people.

- Sunita Ranka (Mumbai, India)

This is fantastic...!

The name "Tea-time with Testers" is itself is so much interesting. Loved this ezine for the quality of its content and the great initiatives you guys have taken up.

Keep up the good work team.

- Avin Hanamsheth (Mumbai, India)

Superb job guys.

Reading about software testing was never a fun before I came across *Tea-time with Testers.*

Great Job. You are surely going to change the picture of Indian Testing Community...!

-Pankaj Parate (Pune , India)

Hi Lalit,

Tea-Time with testers is doing Fantastic job.. You are picking the real time issues and providing solutions....! It is surely helping to all testers to groom their knowledge..!

Cheers !

- Kundan Deotale (Mumbai . India)

You ask... We'll help...!

Anonymous said...

How to connect to excel sheet from vbscript? How can I read the cells and use it for fetching?

Posted on www.tea-timewithtesters.blogspot.com : April 14, 2011 6:56 PM

Dear Anonymous,

Glad to learn that you are referring our blog for help.

Well, Chris has replied to your query. You can check that here.

Would appreciate if you let us know your name and contact information so that we can keep you posted.

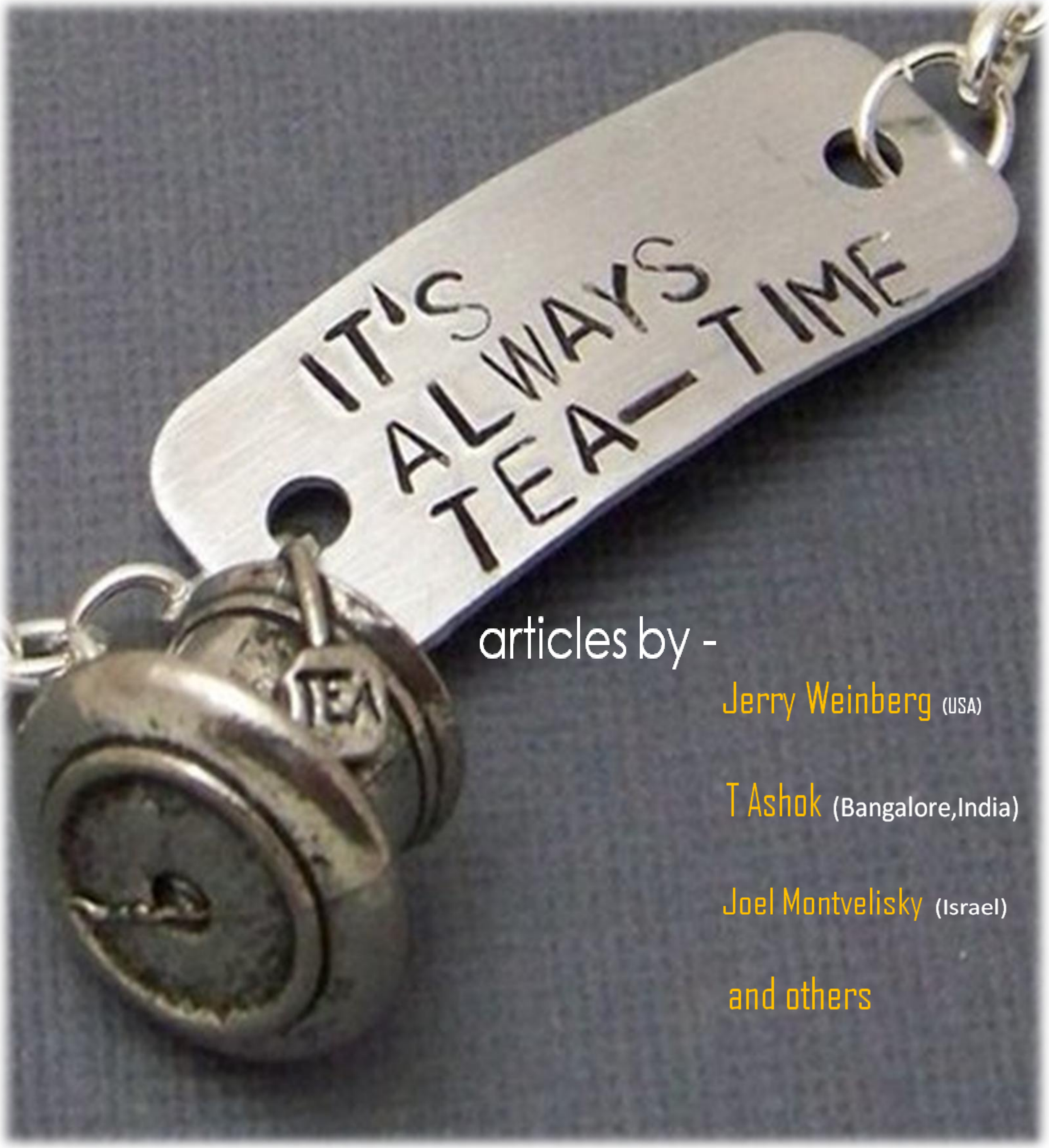
Feel free to contact us for your further queries.

- Editor

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

We are just a mail away: teatimeiwthtesters@gmail.com

in nexxt issue



articles by -

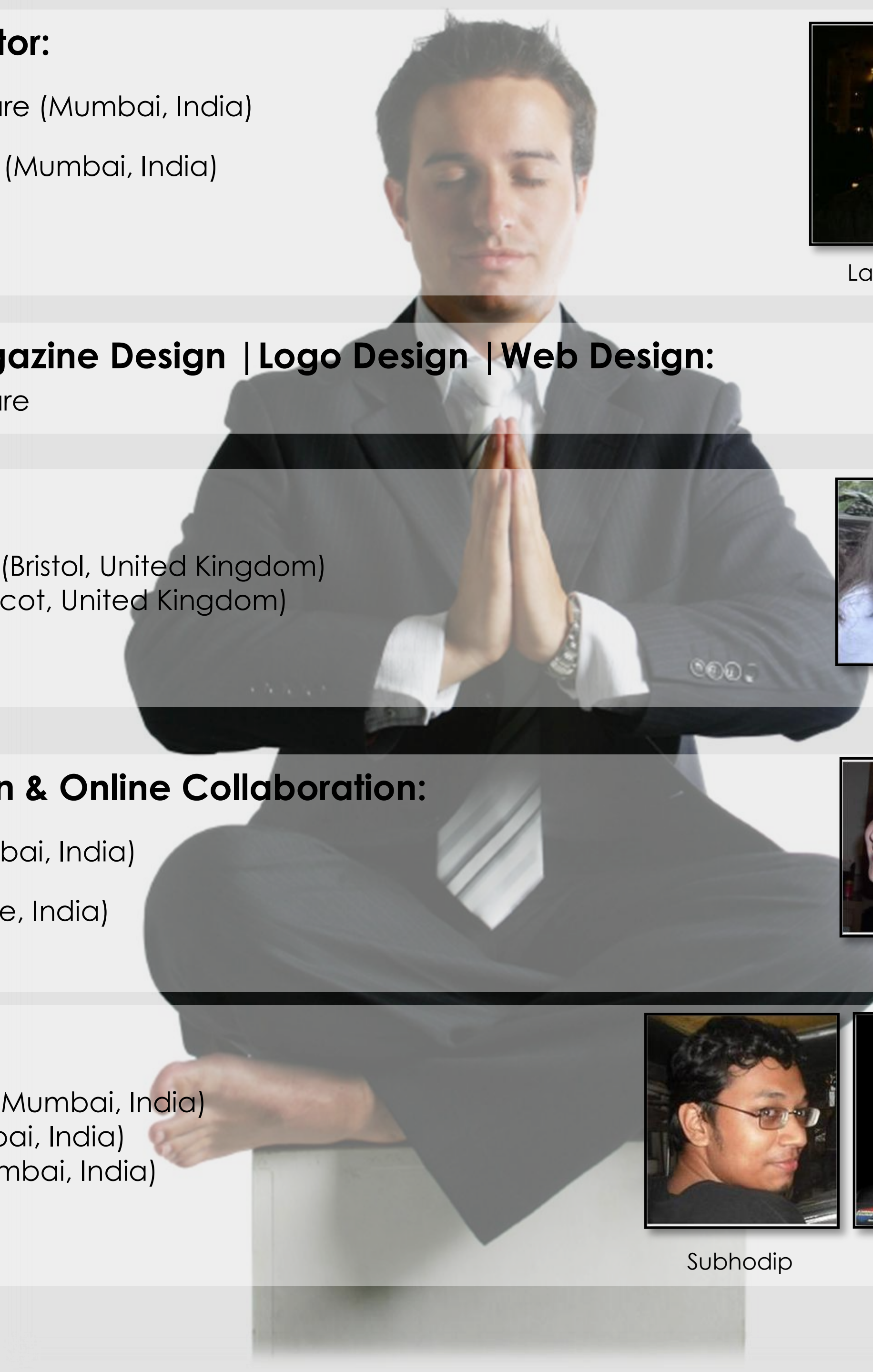
Jerry Weinberg (USA)

T Ashok (Bangalore,India)

Joel Montvelisky (Israel)

and others

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 – Slipcast

Core Team:

Kavitha Deepak (Bristol, United Kingdom)
Debjani Roy (Didcot, United Kingdom)

Kavitha

Debjani

Mascot Design & Online Collaboration:

Juhi Verma (Mumbai, India)

Romil Gupta (Pune, India)

Juhi

Romil

Tech -Team:

Subhodip Biswas (Mumbai, India)
Chris Philip (Mumbai, India)
Gautam Das (Mumbai, India)

Subhodip

Chris

Gautam

|| Karmanye vadhikaraste ma phaleshu kadachna |
Karmaphalehtur bhurma te sangostvakarmani ||

To get **FREE** copy ,

Subscribe to our group at

Google™

Join our community on

facebook

Follow us on

www.teatimewithtesters.com

Give Feedback