THE INTERNATIONAL MONTHLY ON SOFTWARE TESTING

Tea-time with Testers

DECEMBER 2011 | YEAR 1 ISSUE XI

Jerry Weinberg
Measuring Cost and Value

Joe DeMeyer
Basic Application Forensics for Testers

Lisa Crispin
Professional Growth and YOU !

Joel Montvelisky
One Metric = One Big Mistake

T Ashok
Taming Complexity

Mike Talks
The Project Manager's Yellow Brick Road

First Indian Testing Magazine to reach 86 Countries in the world !

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

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

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

Editorial

Dear Readers,

First of all, warm greetings for this lovely season from Tea-time with Testers family.

Packed with all interesting articles as well as some fun to add, here comes the December issue of your most beloved magazine.

Well, designing this issue is very different and equally exciting experience for me. Far away from fast life of Mumbai, currently I am here in my very own village.

With cup of hot creamy tea in one hand and cold breeze playing around, with lovely starry open sky and farms flooded with tempting geosmin, it's just me and this December issue in making. What more one would need when he is doing his most favorite work ?

Shhh…between you and me! There is this big rush at my home, my people are hunting down for me and I am here… secretly doing my favorite job. Why ? Oh…well, I am getting married this week and you know well how tough things become for an Indian groom to be. I might have got wedded by the time this issue lands on your screen ☺.

I hope this issue will give you pleasure as much as it is currently giving me. Feel free to drop me a note if it did ;-).

Anyways I got to run. Let's talk more in our next issue. Needless to mention that it will be flooded with mind blowing articles and some exciting announcements.

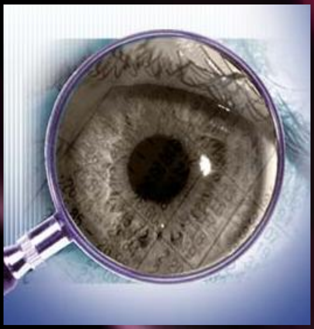
Wish you once again a Merry Christmas and Happy New Year !

Yours Sincerely,

Bhamare

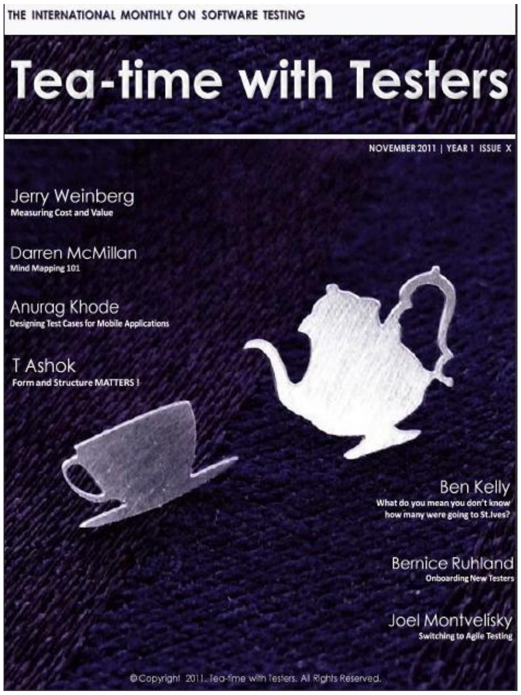
Lalitkumar Bhamare

QuickLook



Testing Puzzles
by Sebi

Crossword
by


THE INTERNATIONAL MONTHLY ON SOFTWARE TESTING
Tea-time with Testers
NOVEMBER 2011 | YEAR 1 ISSUE X
Jerry Weinberg
Measuring Cost and Value
Darren McMillan
Mind Mapping 101
Anurag Khode
Designing Test Cases for Mobile Applications
T Ashok
Form and Structure MATTERS !
Ben Kelly
What do you mean you don't know how many were going to St.Ives?
Bernice Ruhland
Onboarding New Testers
Joel Montvelisky
Switching to Agile Testing
©Copyright 2011. Tea-time with Testers. All Rights Reserved.

November 2011

Have become good tester after reading Tea-time with Testers

Hi Team,

I am Vidhya Shankar. I am a fresher and completed my graduation in 2010. I have done Testing course and currently undergoing training on Automation tool Selenium . I was wondering for interview questions in google and fortunately some how I came across "TEATIMEWITHTESTERS" link …

It impressed me a lot and I downloaded all the Issues from feb2011 to till date and started to read!!!!! Really, I feel that now I have became a good tester and I have that much confidence from all the articles that you have published.

Thank you Team !

LOVE u Tea-time with Testers....!

Thanks to all !!!!!

- **Vidhya Shankar**

My Voice on Teach-Testing !

In current IT market, it's necessary to teach Testing lessons to students in college/universities. The reason behind is whatever we are learning in college and what is recently going on in market are totally different and even not relevant to each other. So when a student comes out from college, it's virtually impossible for him to start carrier in testing. I had also faced same situation.

The other reason is even today there are very few persons around you who will advice you to start job/carrier in testing. Mostly all of them will tell you to learn programming and go for job. I am not blaming that Programming is not the right choice but what if student is not interested in programming language (just like what I am). Luckily I got guidelines from my senior friend about starting with testing. But it's not same case with every student, is it?

The other benefit is that student will know the importance of quality from college when he can learn any new thing very easily. And second this will put an end on those fake institutions, which claim to teach Software Testing for huge money but ultimately teach nothing.

So I just want to vote as YES because whatever I faced, I don't want someone else to face it.

- **Priyank Shah**

Tea-time is the next generation magazine !

Hi ,

I am Carla working as QA manager in L.A.

I have been reading Tea-time with Testers from quite few months and I am impressed with the quality and content you are publishing. I found your publication very unique and different than typical testing magazines. I am not saying that other magazines are bad but the kind of feeling I get while reading Tea-time is really different. Every time I read your publication, it makes me love my field more and more. This is the Next Generation magazine for sure.

Thank You.

- **Carla D.**

TO SEND YOUR LETTERS:

WRITE TO US AT –

teatimewithtesters@gmail.com

What's making News?

- find out the latest happenings in the technology world

mage: www.bigfoto.com

Top six trends to drive market for software testing in 2012

November 29, 2011:

by Manish Shivhare

…….

Companies around the world invest more than $50 billion per year on applications testing and quality assurance, according to Pierre Audoin Consultants (PAC).

Research firms such as IDC and Forrester report a five-year compound annual growth rate (CAGR) of 15.4 percent with spending reaching nearly $19.3 billion by 2015 on testing services alone.

The following 6 trends are in the software testing industry tower:

1) Mobility Application Testing : IDC predicts that the volume of Smartphones is estimated to reach over 500 million shipments by 2014.

These smartphones are used by buyers to congregate their personal and professional necessities. 'There is a plethora of mobile handsets which run millions of applications embedded in it.

Some of the applications today are using the power of cloud to communicate with the ERP systems. In such situations, the data security is of paramount importance. The multitude OS used on the handsets adds to the challenge calling for functionality testing and integration testing.

2) Testing-as-a-Service: Research says that there is increasing usage and demand for TaaS as an alternative for 'avoidable' infrastructure spend.

Today most CXOs perceive TaaS as a tool to address the concerns pertaining to cost reduction, managing wide range of testing projects, test competence needs, exigency to resolve higher defects rates, and using third party's test environment to subside the project requirements. TaaS offers a platform to utilize shared resources in pay as you go model. TaaS is sometimes referred to as Cloud Testing or Managed Testing also.

3) Cross cloud testing: There are very few initiatives taken to standardize the data formats and communication patterns among the clouds. Standardizing the communication patterns will help users to secure their data, gain confidence in their testing vendors and result in seamless transition. Soon we will witness a trend where vendors would engage themselves in testing apps over cross cloud implementation.

4) Business Intelligence Testing: The efficiency of any business intelligence tool does not only reside in its ability to analyze the vast pool of data.

An important indicator of efficiency is how effectively the tool gives meaningful options to its user to extract real time data and identify the key trends using this data. This calls for an extremely efficient system that can pull out the data, clean it and present it in a ubiquitous way without compromising on the performance of the system or causing a downtime.

5) Crowd sourced Testing: Momentum for crowdsourced testing is already building across the industry. IDC expects that traditional outsourcers will increasingly compete with evolving cloud crowdsourcing models for testing applications (e.g., Web apps, mobile apps). As a trend in 2011, it is observed that enterprises now prefer a crowdsourced marketplace so as to get access to global talent and diverse set of skills.

6) Testing catalyzed through test data generation and management: While testing an application, more often than not, confidential information of customers gets exposed to testers.

A breach of this data can lead to serious damage, both to the brand and business. Test data management ensures the availability and security of the test data by obfuscating it on large scale testing engagements. Lots of testing vendors are now researching on various methods to mask this data or create 'test-only' data and help maintain privacy and security while using it.

NASSCOM reported a growth of USD 3.5 billion for testing services in 2010 and a steady growth at 17 percent CAGR until the year 2020. As more and more companies are now outsourcing their testing requirements the market is set for to vault and these 6 towers will be the front runners.

Manish Shivhare works with the Marketing team at AppLabs (a CSC company), the world's largest software testing and quality management company. His responsibilities include Market Research, Analysis and Forecast.

For more updates on Software Testing, visit → Quality Testing - Latest Software Testing News!

Announcing...

Smart Tester of the Month Awards !!!

❖ Winners for Testing Crossword :

Devaganaraja Gopalasetty, Hyderabad
Koushik Rajagopalan, Chennai
Ashwini Sadashivam, Chennai
Dharshini S , Chennai
Abirami G , Chennai
Vemula Krishna , Hyderabad
Saritha S P, Kochi
Jahira Banu A, Chennai
Sonal Agarwal , Noida
Kalyani Muthu, Chennai

❖ Winners for Testing Puzzle :

Sonal Agarwal , Noida

Congratulations !

Discussion helps !

How about talking with us on Facebook?

Come ! Let's have a nice Tea-time there !

Find us on Facebook

CLICK HERE

Look Who's Rising

Eleven Months

10000+ Readers

86 Countries

ONE Magazine

TEA-TIME WITH TESTERS

Subscribe here Right Away to get our all Issues for FREE

Measuring Cost and Value (Part 2)

1.3 The Role of Requirements in Observing Quality

Crosby says quality is conformance to requirements. This is true, if we're careful to speak of meeting the real requirements, not simply matching some written requirements document. In measuring quality, real requirements play an essential role.

1.3.1 Direct measurement of quality

Systems thinking tells us that to produce quality, we must monitor requirements as they change, or as our understanding of them changes. Then we must make adjustments in the process on the basis of deviations between what is required and what is produced.

In other words, requirements provide the standard for direct measurement of quality. At the same time, when we attempt to measure quality directly, we create a process that makes it possible to check the reality of the perceived requirements.

1.3.2 Indirect measurement of quality

Secondary requirements are those not visible to the customer, but related to customer's requirements. Diagrams of effects are used to identify these secondary requirements according to their relationship to the customer's primary requirements. Measurement of quality through secondary requirements is indirect measurement, because it requires an extra step to relate the measurements to quality. And, of course, if there is an extra step, there's just one more possibility of something going wrong.

For example, low "internal complexity" is not experienced directly by customers, who view the software only from the outside. But if internal complexity is high, the customer may experience more failures, more delays to fix failures, more costly fixes, less enthusiasm in responding to enhancements, and possible performance degradation over time. Some of these items are direct requirements, and some are additional secondary requirements that again must be related to primary requirements (Figure 1-1).

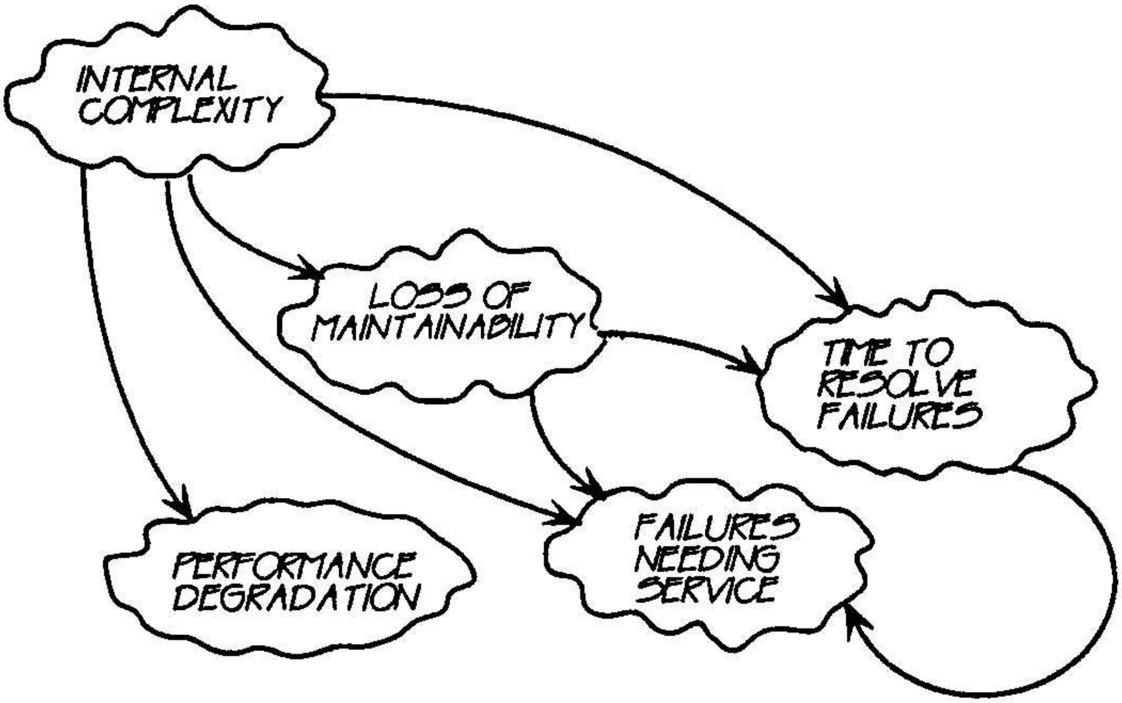


Figure 1-1.

A diagram of effects can trace the relationship between secondary requirements such as low internal complexity and primary requirements such as performance and failures needing service, as well as other secondary requirements such as maintainability and time to resolve failures.

If no relationship can be traced, then the secondary requirement is not a requirement at all. Here's a real example, from a client who builds industrial machinery:

As part of a measurement program, "internal complexity" was being measured routinely by a tool hooked into the FORTRAN compiler. If the complexity was too high, the compiler refused to produce object code. The custom programming group complained that they were spending too much time trying to work their way around this block, but management argued that it was "for their own good." To prove their case, the managers showed the programmers a diagram very similar to Figure 1-1.

The programmers pointed out that most of their work was building "throwaway" programs for the mechanical engineers. If the program was never put into maintenance, then maintainability was not a requirement. Similarly, the service centre would never have to service failures. If the program was thrown away after a single use, performance degradation over time was not a requirement, either.

Ultimately, the conflict was resolved by establishing a procedure for true throwaway programs. These programs did not have to pass a complexity check (though they could if the customer desired). In return, the customer had to sign a document agreeing that the program would be physically destroyed after a certain date. The managers then established a procedure to enforce the trashing of those programs, because they were not so naive as to believe that calling it a throwaway would ensure that it was actually thrown away.

My military advisors, Dawn Guido and Mike Dedolph, assure me that the military is the classic breeding ground of arbitrarily applied standards. Although many of these standards are well thought out, they are communicated in such a way that the rationale doesn't come through, or isn't even offered. The arbitrariness leads to an excess of politics in the interpretation of standards, such as,

a. negotiating for waivers from inappropriate standards

b. repairing the damage caused by inconsistent enforcement of standards

c. complying with standards that hamper work in progress.

For standards to be effective, everyone from managers on down needs to understand:

a. how their job contributes to the goals of the organization (the mission)

b. how their software product contributes to the mission

c. how the standards contribute to the quality of the product

d. how the standards contribute to the success of the mission.

You can get a rather good measure of cultural pattern by the handling of standards. Pattern 4 (Anticipating) managers pay the price to have people trained to understand these questions. Pattern 3 (Steering) managers attempt to answer these questions as they arise. Pattern 2 (Routine) managers typically mandate the standards—though whether they are followed is questionable. Pattern 1 (Variable) managers—if they even have standards—churn around negotiating waivers, repairing damage, and blindly or maliciously complying, but always complaining.

Discontent over standards arises when people who must conform to the standards cannot make the cause-effect connection between the standard and the value of the standard. The less direct the connection, the more the discontent.

Thus, it is useful to have methods of making these connections and assessing their value. In the rest of this article, we will describe two such methods—the detailed impact case method and the single greatest benefit method. Either of these two methods could be applied to proposed standards, for instance, to determine the extent of their value. But they can actually be applied to any situation where quality (that is, value) is important.

1.4 The Detailed Impact Case Method

The detailed impact case method is an exhaustive study of the value impacts of a change. It is based on the idea of tracing requirements through a diagram of effects. Usually, a detailed impact case study is conducted for only a single change so as to ensure accurate observation of small impacts. Of course, several studies can be made, each examining a different change.

1.4.1 Basic approach

The method starts with a brainstorm to produce a list of everyone who could possibly be affected by the change.

Next, brainstorm every possible impact of the change you can think of, to your work or the work of others. For example, consider the effects on the following: what you do, when you do it, what you buy, what you sell, with whom you talk to or don't talk to, how you feel, what your output is or isn't. What you are doing, in fact, is following the technique for developing a diagram of effects, though the participants need not be schooled in this method.

1.4.2 Key ideas

One case is examined in detail to determine value. To keep cost of the survey down, we sacrifice statistical validity for concreteness.

The detailed impact case method is good for tools and methods changes that have a large effect either because they are used on one big project or because they are used with small effect by a great many people, all of whom realize more or less identical benefits. It is also good for cultural changes, for the same reasons.

Any time someone mentions a changed interaction with another person; you must trace it down and interview that person for effects of that interaction. For example, you must follow all changed documents through their life cycle.

What seems trivial to one person may have an enormous effect on another.

Even with each person, several iterations and much imagination may be needed before all effects are listed. Even though you are using one case, you may want to brainstorm with or interview several people who are affected. That way, you can develop a laundry list that includes all possible effects. This list can be reused in similar studies later.

Once you have the list of impacts, you must attach a value to each. This again may take great ingenuity. You can limit the amount of work by guessing which are the big ticket items and doing those first. Stop when you have enough to be convincing, rather than perfect.

to be continued in next issue…

Back To Index

Biography

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

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

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

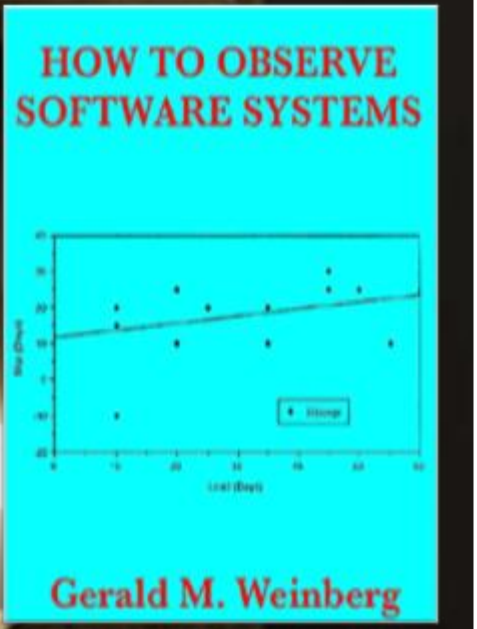
To know more about Gerald and his work, please visit his Official Website here .

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

HOW TO OBSERVE SOFTWARE SYSTEMS is one of the most famous books written by Jerry.

This book will probably make you think twice about some decisions you currently make by reflex. That alone makes it worth reading. "Great to understand the real meaning of non linearity of human based processes and great to highlight how some easy macro indicator can give info about your s/w development process." An incredibly useful book. Its sample can be read online here.

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



TTWT Rating: ★★★★★

Speaking Tester's Mind

- straight from the author's desk

The Project Manager's Yellow Brick Road

By Mike Talks

It's time for the company end-of-year pantomime. But maybe we've been starring in one all year …

Once upon a time, there was a software engineer called Dorothy who worked for a small Kansas IT company. But her career felt too restricting, and she yearned to move to a bigger company. So when she got the offer to up sticks and move over the rainbow to Oz (or was it Aus) she uprooted herself and followed her dreams.

It seemed that someone had dropped a house on her predecessor, and Dorothy had to step into her shoes. Yes they were very nice ruby slippers, and brought with them some considerable power, but with one lost owner they seemed a little cursed.

Dorothy was immediately greeted by a delegation of little people from Oz, who truth be told were glad to see the back of the previous project manager, who they felt had mistreated them. It was all quite exciting, but Dorothy realised she didn't have a team yet.

After meeting with Glinda from HR, she was told to follow the Yellow Brick Road. This was a golden path set out for projects, and the Wizarding CEO at the end expect delivery of a completed project, or would get almighty angry. The Wizarding CEO lived in a gleaming Emerald City - so called, because it provided all the green to fund projects. But the path before them was hard set, and Dorothy shouldn't deviate for any reason.

All looked like it was going to be smooth until the Disgruntled Business Owner Elphaba arrived. She had liked the previous project manager, and was more than a little annoyed to find Dorothy in her shoes instead. She told Dorothy in no uncertain terms to "watch out my pretty".

A little unnerved, Dorothy decided to head down the golden path of project management regardless. The first person she came across was a lion, who was being terrorised by some flying monkeys who were the Business Owners enforcers.

Dorothy managed to shoo the monkeys away, and the cowardly lion was pleased to see Dorothy. He explained his tale of woe to Dorothy. He worked as a Business Analyst, but was too scared by the wicked Business Owner Elphaba and her enforcers. His work was never good enough, and he just kept having to make changes to it.

Sometimes he felt the changes being made didn't make any sense, but he was too afraid to say, feeling it better to do what he was told, and hope to please her one day.

Dorothy took pity on him as he started to cry, and told the Cowardly Lion, if he'd do some requirements analysis for her project, she'd help to keep the flying monkeys of his back. The Cowardly Lion was encouraged by that, and promised he'd follow her.

They went further along the golden path to their objective. The Cowardly Lion had written a few initial requirements down, and was learning to not take it personally when Dorothy made suggestions. They were not complete rewrites, but enhancements, making better and better requirements.

But they soon realised requirements were nice, but they really needed something developed now. They came to a field, to see a scarecrow having the stuffing knocked out of him by more flying monkeys. Dorothy ran after the monkeys, but there were too many of them. To her surprise, the Cowardly Lion let out such a fierce roar that they soon scattered.

The Scarecrow was a sorry sight. They gathered him together as best they could. He worked as a developer, but he was always being told by the flying monkeys how stupid he was. He felt he was too stupid to carry on as a developer he was sure, and was thinking of giving it up and going into something else.

Dorothy told him that was nonsense. They needed a developer to help them, and here he was. The Scarecrow wasn't sure at first. He finally agreed to take it on, but only until they found someone more suitable and cleverer.

So they set out again. This time Dorothy didn't seem to have much to do. The Cowardly Lion passed the Scarecrow his notes, and talked them through a bit at a time, so the Scarecrow wouldn't get overwhelmed. Every so often they got to something they weren't sure of, and Dorothy would make a decision on what she thought would be right. But mostly they could sort it out between them.

The Scarecrow seemed to forget his initial reluctance, and got designing and coding, asking questions of the Lion as he went along. The code seemed to be coming together, but they couldn't really be sure. What they needed now was a tester.

As luck would have it, at this point they came to a metal man, who'd rusted into a statue. He made a plea, but no-one could work out what for. It was the Scarecrow who noticed that in the metal man's hand was a piece of paper which read,

Category 1 incident

I have noticed I am starting to rust, and require immediate lubrication, as I'm continually losing motor function as the corrosion continues.

The Scarecrow worked out that maybe the metal man needed oiling. He found an oil can nearby, and together he, the Lion and Dorothy oiled and pulled until the metal man could move again. He said he was called Tin Man, and he worked as a tester. Unfortunately he'd not got on well at his last project, and was told he raised too many defects. In fact his co-workers told him he didn't have a heart with some of the things he came to them with. So they'd left him outside to rust. He'd tried raising a defect report to get oiled, but no-one helped.

Dorothy couldn't believe her luck, just as she needed a tester she found one. But she was concerned - the Lion was only starting to find his courage, and the Scarecrow was only starting to believe in himself. A tester without a heart could set them back all the way to Munchkinland.

“Look” she asked the Tin Man, “we have to get this project as finished as possible before we see the Wizzarding CEO in the Emerald City. I need you to really look for the big things, and try not to make too much fuss over the little ones”.

“Oh” said the Tin Man. He was quite taken aback. Usually it was his job to point out all the faults in other peoples work, and he was very good at it. But what Dorothy said made sense to him, the big issues were the important ones, and he'd tackle them first, but list the little ones for later reference.

Even so testing did not go well. The Tin Man asked questions of the Lion, who became afraid and lost his voice. When the Tin Man noticed problems in the Scarecrow's code, the Scarecrow said this was proof that he didn't have a brain, and shouldn't be doing this.

But it did get better – as the Scarecrow worked on the bugs, the software became more robust, and the Tin Man remembered to tell him this, and how good the new builds were. The Scarecrow even started making suggestions to improve the design.

The Lion began to realise the Scarecrow and Tin Man were deferring to his judgement and calls about how the requirements should be interpreted and became more confident. And the Tin Man began to realise he was a valued equal member of the team, and felt they were all working together to improve quality.

The wicked Business Owner tried many time to take them off course, to send them in the wrong direction or to send them in circles. But Dorothy managed to keep them on the right path and pour water on their problems.

By the time they reached the Emerald City, their work was done. The Wizzarding CEO was much impressed, and he told them as much in his teleconference.

The Lion had shown his courage, the Scarecrow his brains and the Tin Man his heart. And it was all down to Dorothy's leadership.

The Wizard of Oz is obviously a classic film. What's interesting is that though Dorothy is the heroine, she of herself doesn't do anything that amazing. Though she defeats both witches, it's more by accident.

However in her journey she meets a dysfunctional group of people in the Scarecrow, Tin Man and Lion. But through her mentoring and leadership, they become capable of much greater things.

This is a quality of truly great managers. Thus they are shoes we should all aim to fill …

Mike Talks is a Senior Tester in Wellington who's worked on a number of projects around New Zealand.

A tester who likes to make a song and dance about everything - he usually hogs the microphone on company kareoke nights.

And yes, he was in his High School's Musical performance of The Wizard of Oz, where he actually was dating the Wicked Witch …

He posts on Twitter as TestSheepNZ .

: Professional Growth:

is it unrealistic to do it on your own time?

By Lisa Crispin

I spend a lot of my own time writing, reading, attending conferences, participating in local user groups, joining in Weeknight Testers, Code Retreats, testing dojos.

My experience is that the more I practice my professional skills, the more value I can contribute to my team and company. Even more importantly, it brings me more joy in my work. I go around urging others to invest their own time in their professional growth.

In his book *The Clean Coder*, Uncle Bob Martin asserts that a true software professional spends 20 hours a week of her own time practicing and learning her craft. When I mention this to people, I often get a similar reaction that I see in several of the reviews of *The Clean Coder* on Amazon: "If I don't spend 20 hours a week reading and doing katas, am I not a professional?", "I have a family, there is no way I have 20 hours a week to do that.", "If my employer wants me to learn more, they should give me time on the job."

Working at your avocation

I get that everyone working in software does not feel their job is their true vocation. They might simply be paying the mortgage while they pursue their artistic career or raise their kids. But if software development is something you truly enjoy, you'll enjoy it more as you become better at what you do. You also are likely to have a more successful career if you constantly learn, not to mention the benefits of networking through many of these learning opportunities.

For awhile, I puzzled over why there's so much antipathy to the idea of spending your own downtime doing the same sorts of things you do at work. If you like what you do for a living, why wouldn't you enjoy doing it outside of work as well? I think coding and testing are fun! It's not punishment to work at it on my own time. If I were, say, a professional musician, I'd have to practice on my own time. What's the difference?

I finally have a theory as to why many people in software are outraged by the notion they should be doing something work-related on their own time. I could be wrong, but I think this theory has possibilities.

I can't swear that I spend 20 hours per week practicing my craft, though it might average out to that over a year. I usually spend a couple hours of my own time per day reading online articles and blog posts, and writing my own blog posts and articles. I use the bulk of my vacation time attending conferences, and I spend a ton of my time preparing tutorials and talks for these conferences so that I can attend for free. I go to some events on my own dime, for example, I spent my own money to attend Agile Coach Camp this year.

Why do I do this? It's fun! I have many friends in the agile community, and I love learning. For a long time I thought I would work in software until I figured out what I wanted to do when I grow up. One day I realized I was actually passionate about delivering a high quality software product for a customer.

My theory

So how do I have time for a family life and my many other interests, such as riding horses and driving donkeys, and still spend so much time on my own professional growth? Well, at my actual job, I work at a **sustainable pace**. Yes, my teammates and I work about 40 hours a week. And that 40 hours per week, on average, includes significant slack time to learn and experiment while at work. Our management understands the value of taking time to keep technical debt at a reasonable level and finding ways to innovate and improve. We take time to do things right. We enjoy our work. What a revolutionary idea!

I meet many people who say they don't have the luxury of working a 40-hour week. They are forced to work at a frantic pace all the time, and they often have to work late or on the weekends. Well, yeah, if I worked in that sort of environment, I wouldn't have the energy to practice my craft on my own time, either. Though I think such a situation might motivate me to acquire new skills, so I could find a better place to work.

If you don't have time on your own to devote to professional development, you might want to take a look at your job. It might be a chicken and egg thing. If you don't have the skills and experience that are in demand, maybe you can't get a job with that awesome agile team that actually works at a sustainable pace. If you work in a dysfunctional environment where the team is always on a death march, you may wish to change jobs but you're too exhausted to try to acquire the necessary skills and experience outside of your regular job.

Back in 2000 when I joined my first agile team, I loved it so much that I wanted to make sure that good job opportunities on other agile teams existed. That's one reason I've worked so hard to spread the joy of agile and help people find good ways of delivering value frequently at a sustainable pace (to paraphrase Elisabeth Hendrickson). Agile is now mainstream. That doesn't mean every shop that says it's Agile that has teams working at a sustainable pace, but I do see more and more teams who "get it".

Be the change

I'm a big believer in baby steps. If you don't do anything outside of work for professional development, and you're so busy at work that your personal life would suffer if you "worked" on your own time, why not try setting aside just two hours a week to join in a Weekend Testing session, read a good testing magazine (oh, you are reading one now!), go to your local testing user group meeting, read a great job-related book. You may find yourself inspired to join in more learning and networking opportunities. These could even lead you to a position where you don't always have to work like a slave.

In my view, when someone like Uncle Bob Martin recommends 20 hours a week for personal professional growth, it's a tool, not a rule. There's nothing magical about the recommended amount of time. The important thing is that you invest in continual improvement.

Companies that are successful in the long term don't consistently overwork their employees. Consider being a change agent in your own organization to start chipping away at technical debt and finding room for professional development both at work and outside of work.

It's really not crazy

Nobody is asking you to neglect your family or turn into some kind of software drone. We're passionate about what we do, and we just want you to enjoy your jobs as much as we enjoy ours. (And by the way, even I have days where I think about changing careers. Nothing's ever all daffodils and butterflies!)

I hope this doesn't sound too sanctimonious. If you're just marking time in the software business to pay the bills until you can switch to your true avocation full time, invest your time in what you really want to do. If you want to be part of a top-notch software development team and look forward to going to work every day, invest some of your own time to get there.

Back To Index

Lisa Crispin is the co-author, with Janet Gregory, of *Agile Testing: A Practical Guide for Testers and Agile Teams* (Addison-Wesley, 2009), co-author with Tip House of *Extreme Testing* (Addison-Wesley, 2002) and a contributor to *Beautiful Testing* (O'Reilly, 2009). She has worked as a tester on agile teams for the past ten years, and enjoys sharing her experiences via writing, presenting, teaching and participating in agile testing communities around the world.

Lisa was named one of the 13 Women of Influence in testing by Software Test & Performance magazine. For more about Lisa's work, visit **www.lisacrispin.com**.

Lisa can be contacted on Twitter @lisacrispin .

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

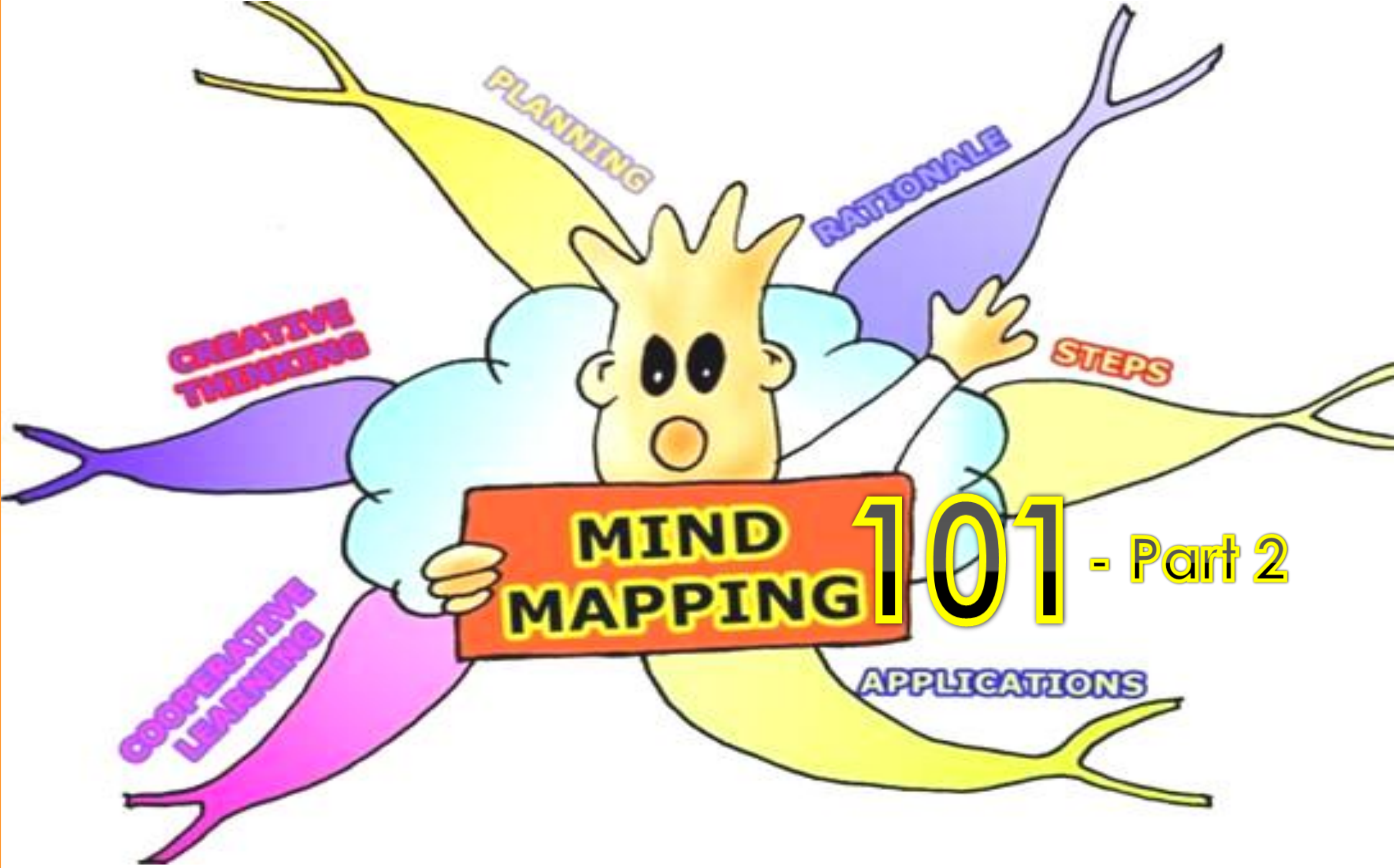
No Problemo! We will publish your Feedback/Comments and also the answers to your Questions that you have for our Authors.

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

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

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

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



MIND MAPPING 101 - Part 2

By Darren McMillan

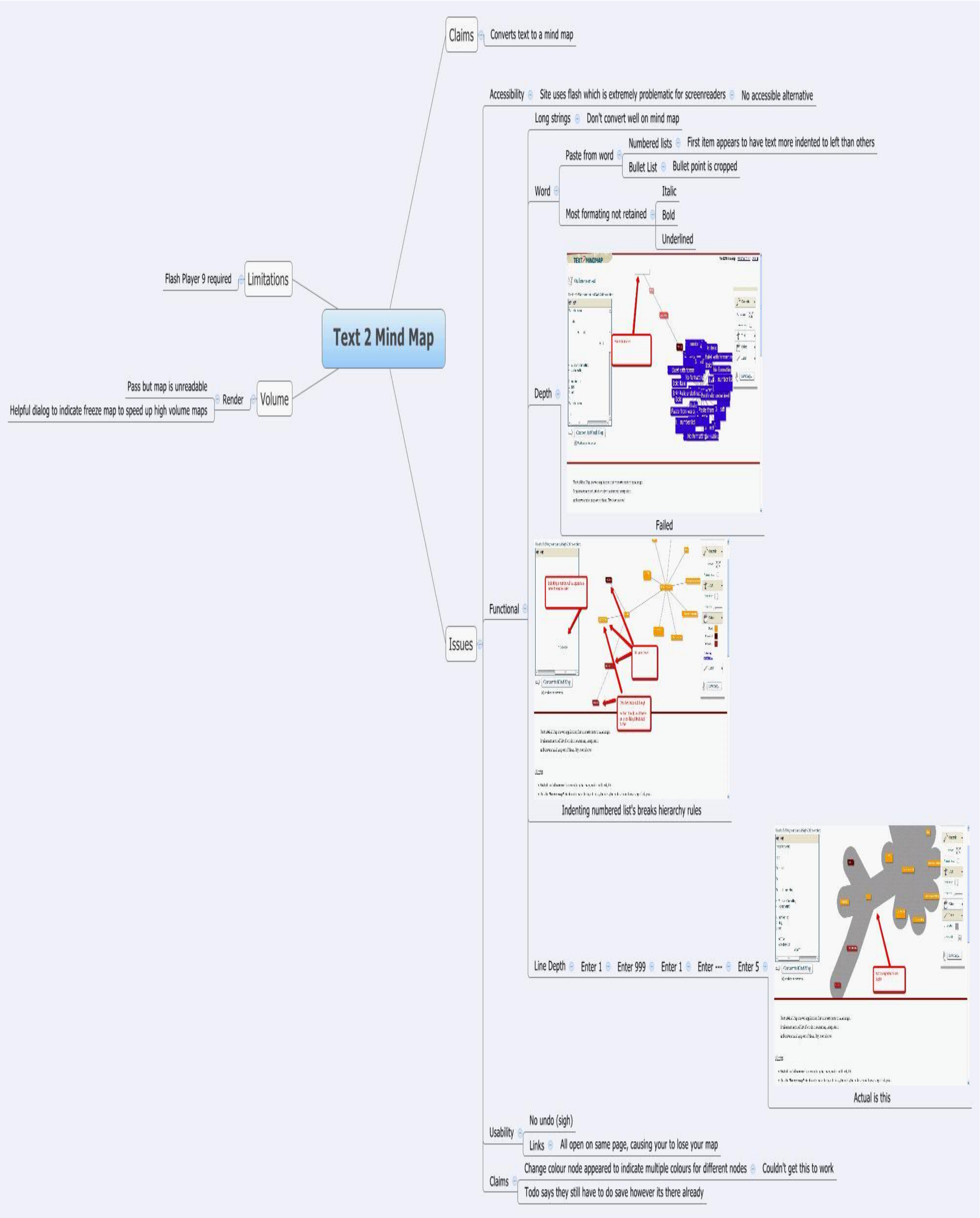
Session Reports

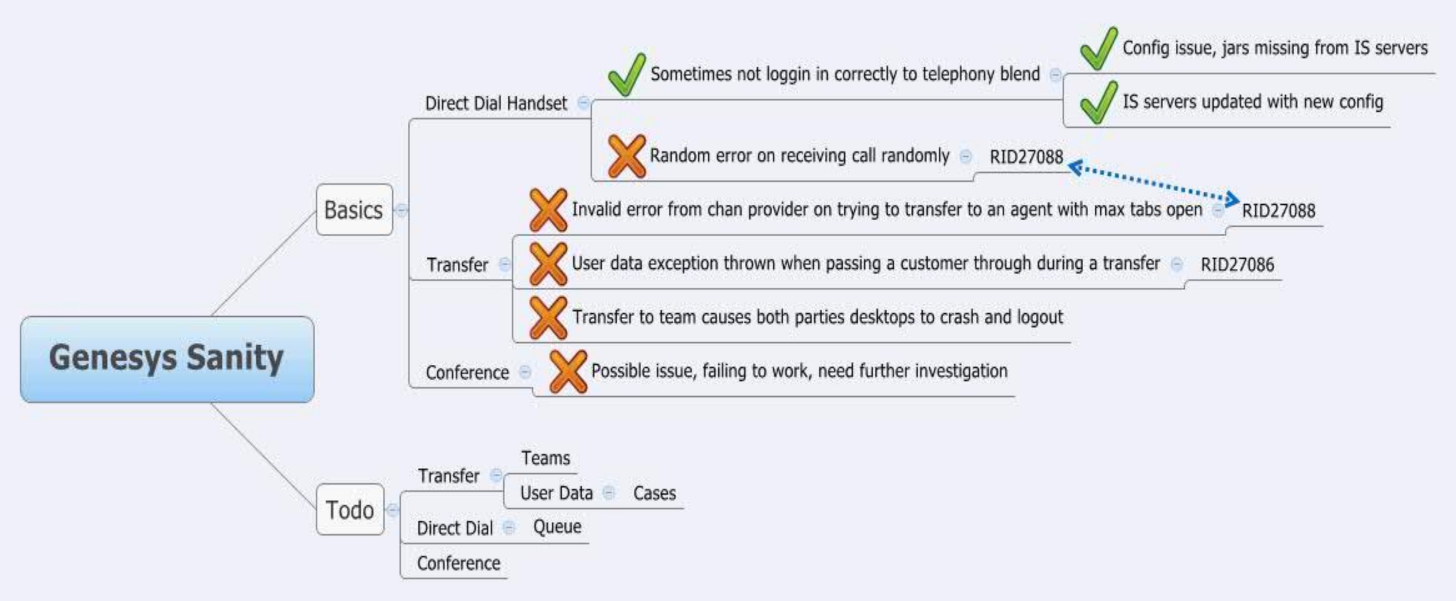
Mind maps can also be useful for feeding back test results or the progress of a testing task.

An example of a testing session report using a mind map is provided on page 31, from when I attended a weekend testing session and was asked to test a text to mind map tool.

You might even just want to feedback the progress of a testing task in an understandable format, without having to write extensive reports. The map on page 32 is an end of day report on a late test phase task inclusion for a telephony provider sanity check.

(Note: Please click on all images to enlarge them)

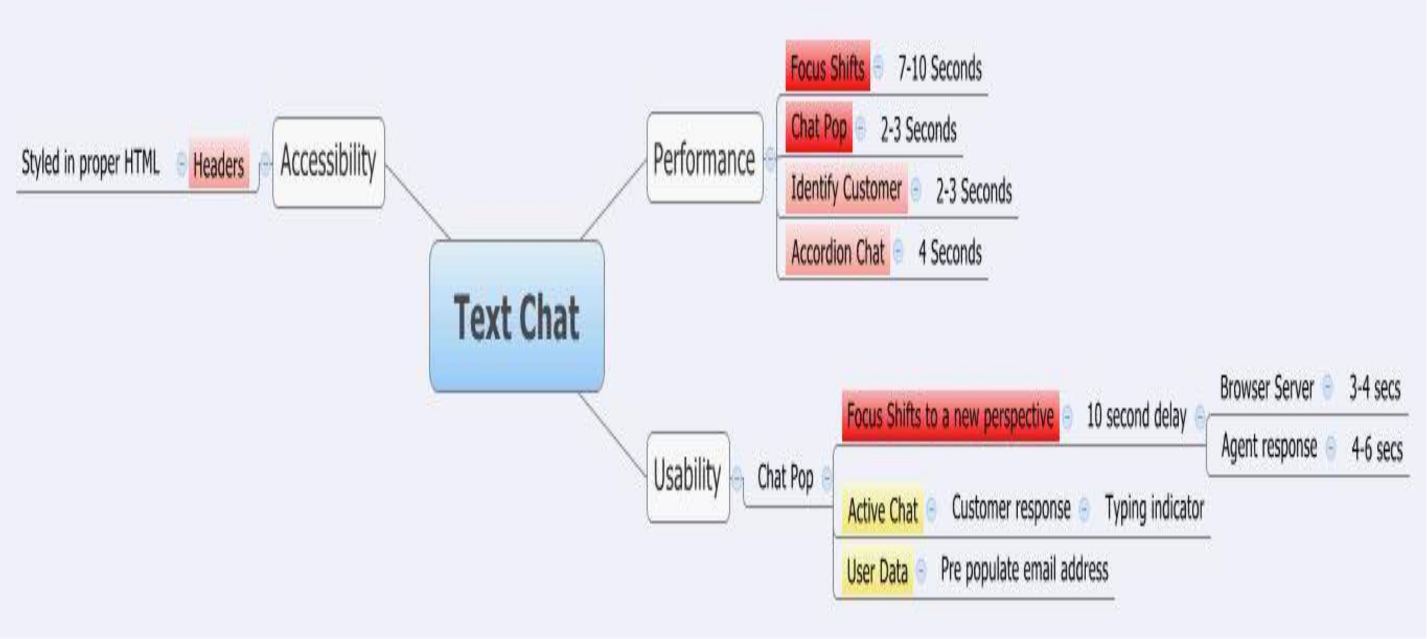

Text 2 Mind Map
Claims
Converts text to a mind map
Limitations
Flash Player 9 required
Volume
Render
Pass but map is unreadable
Helpful dialog to indicate freeze map to speed up high volume maps
Issues
Accessibility
Site uses flash which is extremely problematic for screenreaders
No accessible alternative
Functional
Long strings
Don't convert well on mind map
Word
Paste from word
Numbered lists
First item appears to have text more indented to left than others
Bullet List
Bullet point is cropped
Most formating not retained
Italic
Bold
Underlined
Depth
Failed
Indenting numbered list's breaks hierarchy rules
Line Depth
Enter 1
Enter 999
Enter 1
Enter ...
Enter 5
Actual is this
Usability
No undo (sigh)
Links
All open on same page, causing your to lose your map
Claims
Change colour node appeared to indicate multiple colours for different nodes
Couldn't get this to work
Todo says they still have to do save however its there already


Genesys Sanity
Basics
Direct Dial Handset
Sometimes not loggin in correctly to telephony blend
Config issue, jars missing from IS servers
IS servers updated with new config
Random error on receiving call randomly
RID27088
Transfer
Invalid error from chan provider on trying to transfer to an agent with max tabs open
RID27088
User data exception thrown when passing a customer through during a transfer
RID27086
Transfer to team causes both parties desktops to crash and logout
Conference
Possible issue, failing to work, need further investigation
Todo
Transfer
Teams
User Data
Cases
Direct Dial
Queue
Conference

Simplifying feedback to management

Many often find the process of reporting information back to management types difficult. I'm not different. I have spent a long time trying to find a method which worked for all. After time I found that by simplifying information feedback into a mind map with levels of importance highlighted on it I'd have much better outcomes from these meetings with management.

One example was a low bandwidth test we'd done with our Indonesian based test team. I was playing the role of a call centre agent and they were acting as customers. In a 60-90 minute session we managed as a by product to get some performance and usability feedback. This was communicated to management in a quick fifteen minute meeting using the mind map below. All but one item was followed up on, which is much better results than normal. I guess because they found the information easier to digest and as a result my job of firming up why these issues should be looked was simplified.

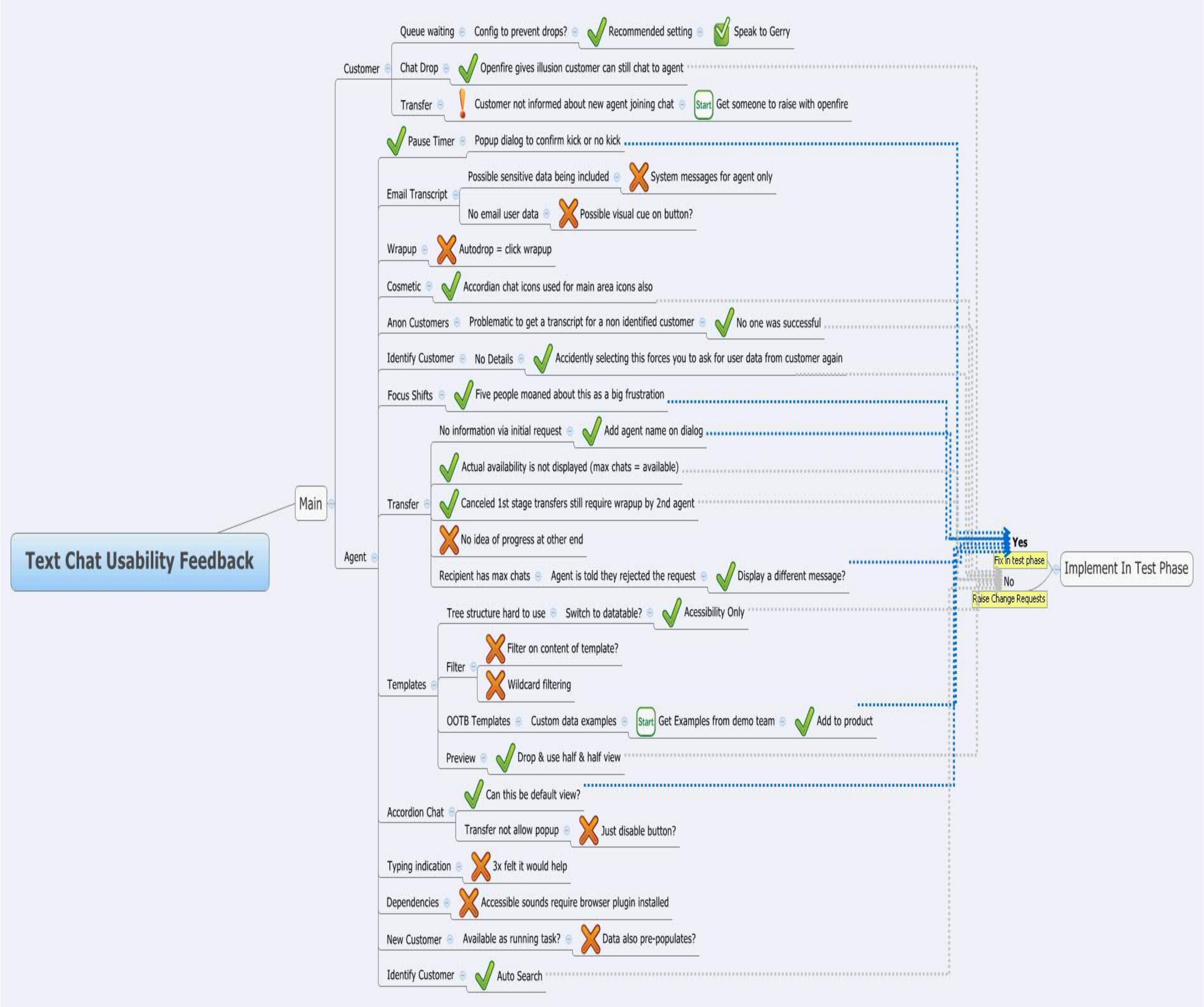


Simplifying feedback to groups

Sometimes you might have quite extensive information that you'd like to feedback to a group of people. Mind maps can be fantastic for this as they allow you to simplify that information and convey it in an understandable format that others can digest quickly.

One example I can provide was from a recent usability evaluation I did. The information had to be conveyed to three different teams and management. It had to illustrate what had happened with the feedback provided from that usability evaluation in a simple understandable form.

You can see from the mind map below that I did manage to achieve that and even got responses from a few people later including the product manager thanking me on the use of a mind map to convey this information.

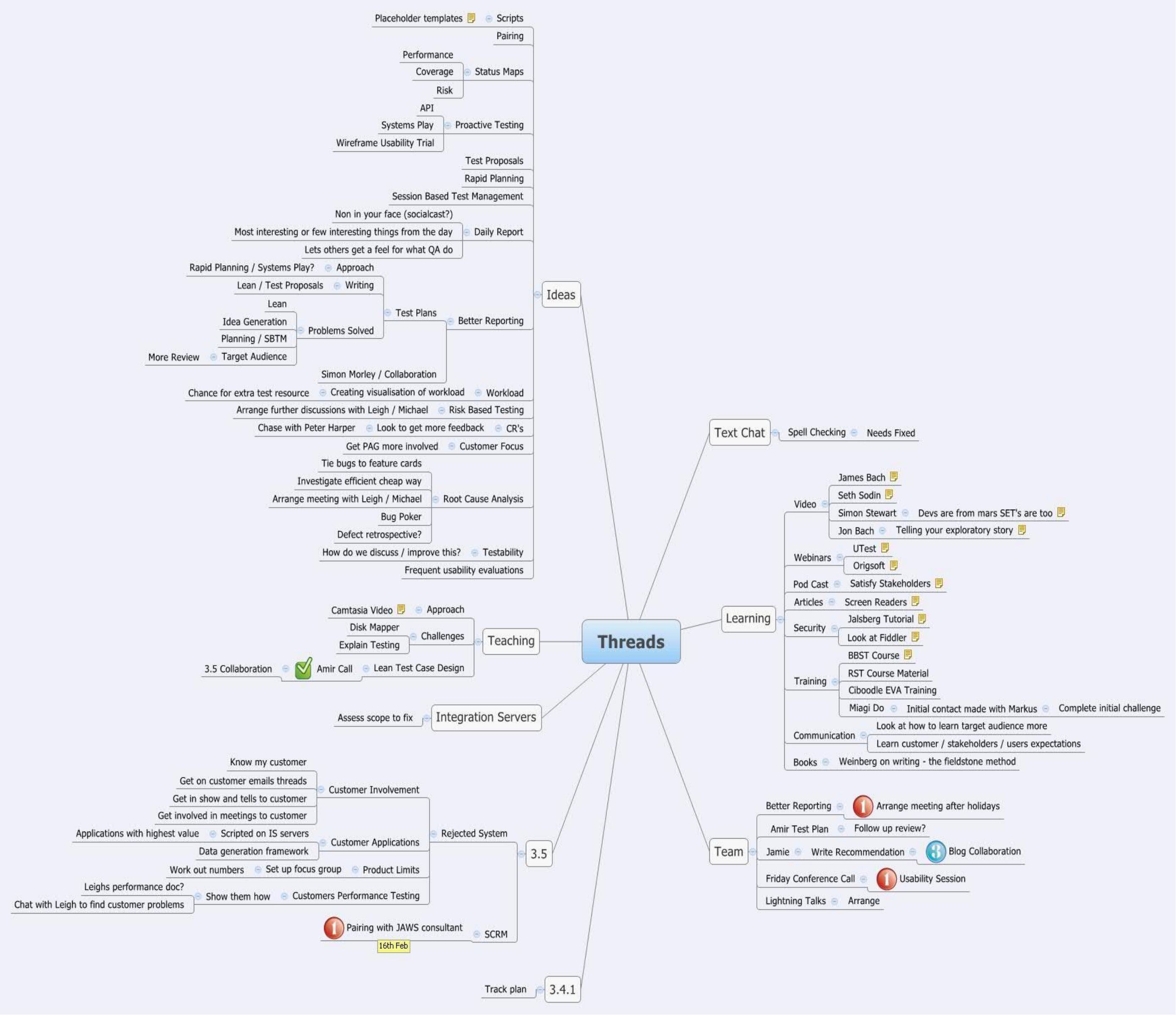

Text Chat Usability Feedback
Main
Customer
Queue waiting
Config to prevent drops?
Recommended setting
Speak to Gerry
Chat Drop
Openfire gives illusion customer can still chat to agent
Transfer
Customer not informed about new agent joining chat
Start
Get someone to raise with openfire
Agent
Pause Timer
Popup dialog to confirm kick or no kick
Email Transcript
Possible sensitive data being included
System messages for agent only
No email user data
Possible visual cue on button?
Wrapup
Autodrop = click wrapup
Cosmetic
Accordian chat icons used for main area icons also
Anon Customers
Problematic to get a transcript for a non identified customer
No one was successful
Identify Customer
No Details
Accidently selecting this forces you to ask for user data from customer again
Focus Shifts
Five people moaned about this as a big frustration
Transfer
No information via initial request
Add agent name on dialog
Actual availability is not displayed (max chats = available)
Canceled 1st stage transfers still require wrapup by 2nd agent
No idea of progress at other end
Recipient has max chats
Agent is told they rejected the request
Display a different message?
Templates
Tree structure hard to use
Switch to datatable?
Acessibility Only
Filter
Filter on content of template?
Wildcard filtering
OOTB Templates
Custom data examples
Start
Get Examples from demo team
Add to product
Preview
Drop & use half & half view
Accordion Chat
Can this be default view?
Transfer not allow popup
Just disable button?
Typing indication
3x felt it would help
Dependencies
Accessible sounds require browser plugin installed
New Customer
Available as running task?
Data also pre-populates?
Identify Customer
Auto Search
Yes
Fix in test phase
No
Raise Change Requests
Implement In Test Phase

Self Organisation

In the past I've used many tools to manage my schedule and tasks (my to-do list essentially). None every suited my needs fully, and required either too much time to maintain, or provided poor visibility of my scope.

It dawned on me that mind maps are not only easy to maintain, but also provide high visibility. As such I converted my to-do list over to a mind map, and have never looked back since. I can add, remove nodes quickly when new items come into scope or old are complete. I can also highlight and mark dates on my priorities to aid visibility.

You can see my to-do list in the mind map below.



Others

There are other things I use mind maps for that I don't currently have examples of.

- Requirements Analysis
    - Gaining a quick understanding of proposed requirements
    - Pin pointing gaps, conflicts.
- Throw away
    - Quick analysis of a feature
    - Determining if an idea is achievable or not.
    - Test coverage for a feature
- Mentoring
    - Using mind maps collaboratively to design tests with others.
    - Generating test ideas with others.

Most mind maps in this document were created using the free edition of XMind.

Thanks for reading.

Back To Index

Discuss this topic on Quality Testing. Click HERE !

Darren McMillan has been working in the testing field for over four years now. He has a genuine passion for testing all things and actively seeks to solve the problems others tend to accept. He strongly believes that opportunities are there to be taken and actively promotes self learning to others. When he is not testing or writing about his experiences, he enjoys nothing more than some quite family time.

A proud father to a beautiful daughter he hopes that from leading by example he will encourage her to follow her own dreams. Contact Darren on Twitter @darren_mcmillan.

Basic Application Forensics for Testers

by Joe DeMeyer

Your conversation with the developer begins easy enough. They run through your scenario as you described, they ask some clarifying questions, you respond with what you observed and what you did. They execute your instructions and the defect you discovered appears.

The developer sits back with a "How'd that happen?" look and turns to you. They talk about a specific application function and display the code on the screen. They explain how it works and its business purpose. This time, you ask clarifying questions about a decision branch and the execution of a function in the middle of the branch. Their reply begins with a phrase like "Under the hood,…".

While you appreciate the technical decision to implement a function like this, you think to yourself testing will be difficult or time consuming. You've seen this before: infrequently executed operations, complicated test case initialization, asynchronous processes, complex logic or algorithms, or long-running programs all require time and effort to evaluate. In these scenarios, it is difficult to locate direct evidence of both success and failure. The testing effort compounds when you find defects and a scenario must be re-created.

Opaque operations reduce the testability of an application and raise the cost of testing. These costs include time to set up the test (more time where there are multiple teams participating), time to locate evidence of execution (pass or fail), and time to re-create the scenario to determine if a defect has been resolved. Costs increase for application modules that are difficult to access even through structural methods. Testing difficult programs also has an impact on its users when it is not available or not working correctly.

The introduction of structural methods into your testing strategy begins to provide not only transparency to these operations but information about them sooner in the project cycle. In that light, transparency is central to testability. In this article, I explore basic software forensic methods which make program operations transparent with little or no impact on the application.

Journaling

Journaling is the easiest application transparency method. Many developers already use this method to assist with troubleshooting. They place statements in the code which write contextual information (a variable's value, the executing line of code, or just a status) to a buffer, screen, or file during program execution.

This information tells the story of what the code actually does. Once you learn what you need, the statements are easy to remove. However, if this is the only transparency method you use, focus your effort on a well-designed, consistent format. It can become a business record useful in resolving non-production and production defects, customer questions, or legal issues.

As A Tester...

When you want to understand what an application does or want to verify a defect, have your developer insert statements into the code to provide transparency to its operations. After you both have collected the information about the operation, remember to remove the statements.

Error Injection

Error Injection acts like a contrast agent in a program. If you have edited pictures on a computer, you may have lightened or darkened an image to improve its appearance. This is a change to a picture's contrast. Similarly, the introduction of an error into a program serves to highlight code that is difficult to evaluate, or whose execution is difficult to verify.

Error Injection is a method of purposefully causing a program to fail in an effort to learn more about its operation. Injected errors contrast well against normal operating code and highlight opaque operations or calculations. Note: This method of transparency must be applied carefully and with the knowledge of the project team.

You can cause a failure in most programming languages by dividing a variable by zero. Insert a line of code to divide a variable by zero and start the program in debug mode. The program stops when it encounters the injected error. Alternatively, changes to calculations or displayed text may also reveal the path to code that is difficult to test. Once the path is discovered, remove the injected error.

As A Tester...

When an operation or calculation is difficult to locate or evaluate, collaborate with your developer to inject errors.

Web Services

A web service is used to exchange formatted information between machines on a network. A web service waits for a request, processes it, and responds with data or status. Web Services present challenging test scenarios because there is no user interface. However, both the web service request and response may be written to disk to evaluate the operations of the web service. The format is XML and corresponds to an interface contract for the web service (this contract is known as Web Service Definition Language or WSDL).

As A Tester...

When you begin testing a web service, ask your developer to write both request and response to disk. A little research may help set up test cases (such as handcrafting requests for success and failure), provide evidence of execution, and assist with troubleshooting for both testers and developers. For example, if the web service processes information based on a date input, handcraft many requests with different dates that exercise web service processing.

Database

Many business applications execute the life cycle of their business objects in a database record or records. Data is collected, changed, dated, or assigned a status. A database is a common repository of business data and a common artifact for validating application functionality.

While database tools provide views of data, custom queries are valuable for sorting and filtering. In all phases of a project, create or request custom queries to isolate and report on application interaction with the database.

Some common queries are for status field values or changes to important data. Other useful queries return the number of records during a certain time period, specific field values, or time and date sensitive records. These reports supply evidence of correct and incorrect functioning in your application. Over time, a library of custom queries serves as regression check points, and learning tools for future queries.

As a Tester...

When you begin a project where database changes are planned, learn about the database by writing queries. As you learn more about the changes, tune your queries to provide information with and without the changes.

For example, if the project adds new columns to the database, use a SELECT query to return sample rows with all columns. Save this result and execute the same query after new columns are added. The difference between query results should be the new columns.

For another view of database operations, a database trace tool provides detailed information around changes or stored procedure execution. When you request a trace, request specific information and limit the trace duration. A trace requires both processing time and disk space. When you focus the purpose of your trace, your database administrator will appreciate the conservation of both.

Errors

Applications generate errors. While some errors are defects, many provide information about the program or are a focal point for more investigation. In many cases, an error results from a condition which may be beyond the scope of the application's purpose. For any type of error, have the development team create an error dictionary. Include the error's definition, how it is caused, and how it may be resolved. Also, assign a unique error code for easy identification.

Popular operating systems support an intrinsic repository where errors are logged. They can be used to record errors for specific applications. When configured for this repository, the date, time, error messages, and stack dumps may all be part of the error log. Table 1 lists these repositories and their corresponding operating system.

As a Tester...

Become familiar with system utilities that report on errors and other events on the machine. Collaborate with your team to have your application log errors to a common repository to save time in locating them, and provide information for troubleshooting.

Documentation

Application documentation is a good source of transparency when it is available. This includes requirements, design documents, the program itself, production support, and previous defect reports. Of all these documents, the program and production support are usually the most recent and most accurate.

As a Tester...

Production support documentation defines possible failure modes and their resolution. A review of these documents may be a source of test ideas for a new project, and motivation to create new or edit existing documentation.

The Program

The program itself defines the application's behavior (though not always the expected behavior). It is the most direct form of transparency. Not only a source of documentation, you can use it to generate test ideas. Additionally, a source code repository (where programs are maintained) may also provide test ideas. Review the source code repository for check-in history and check-in velocity. Source code files that are checked in often may be candidates for more inspection.

As A Tester...

You benefit from reading the code because it serves as a code review, a structural test, and prompts ideas for test cases. Alternatively, attend code reviews and design reviews to learn about application structure and developers' concerns, and use them to generate test ideas.

Tools

Operating systems provide information about applications in the form of administrative tools. Additionally, there are many third-party tools available at sysinternals.com. The most common operating system tools collect system, status, and error messages in a single place. A review of this repository may reveal information about your application.

If your application logs information about its operations, consider creating a tool or tools to collect and process the information. These tools provide an opportunity to centralize the information from logs and, where possible, create a common reporting format. Of course, you must consider maintenance of the tools you build and balance it with the benefit received from the reports.

Lastly, a coherent report may provide not only transparency but a profile of your application. This kind of baseline view can be used to improve and measure your application.

As A Tester...

The creation of tools for the purpose of transparency can aid in reducing costs. Where evidence of execution becomes easier to demonstrate and validate, the cost of testing is lower. Become an advocate for transparency by watching for opportunities to build these kinds of tools.

Transparency Planning

Many of the methods described above can be implemented in an ad hoc fashion and provide a temporary benefit of revealing subtle operations of your application. This is an excellent way to pilot transparency in your application and determine its benefits. However, with some planning, application transparency can provide longer term benefit for project, business, and legal stakeholders. Here are some considerations for implementation.

- Match the Method to the Task

When you think about the purpose of the information you want, ask yourself which method best suits my purpose, and how the information will be used. Journaling is useful during the initial and middle stages of a project where code is under development. It is easier to implement and provides information quickly however it may be difficult to maintain.

Using a database requires some design considerations during project planning and some thought around its integration into the application. Note that it provides benefit for not only you as a tester, but developers and stakeholders as well.

Introduce transparency mindfully and purposefully. Start simple and reflect on the information you generate. Use what you learn to grow the design and to grow the amount of information produced in a disciplined manner. Poor implementation or rapid introduction of transparency methods can impact application maintenance.

- Have a Contingency

An application which uses a database usually has a contingency plan if the database becomes unavailable. For example, some applications will write information to disk. When the database is working again, a batch process reads the information from disk, formats it, and populates the database.

If the method of application transparency fails, think about having a contingency plan. Consider the cost of information loss against the benefit the transparency provides.

- Control the Transparency

Regardless of the method, transparency generates information. For those methods that store the information to disk, consider using a switch or throttle to control when or how much you store. This suggestion may also assist with successfully introducing transparency.

A switch may be used to turn your transparency off and on. It typically resides in an application's configuration file where changing a switch's value from "true" to "false" or "on" to "off" is simple and quick.

A throttle may be used to control the amount of information generated and can also act as a switch. Typical throttle settings are Off, Simple, and Verbose (you may need different settings). Off stores no information, Simple stores basic information, and Verbose stores all available information. Similar to a switch, a throttle value may be stored in a configuration file.

Transparency Maintenance

Let's say your project team wants to introduce transparency into the application. They want to store the information in a database and plan how to format and analyze it. After the analysis, what should happen to the collected information?

A couple of choices are to archive it or to dispose of it. An archive of the information insures having it for review or investigating it for some new purpose. In some industries, there is a legal responsibility to maintain information for a period of time. Disposing of information makes sense when it is useful for the instance it was created.

Table 1. Operating System Transparency Tools

Windows	Event Viewer
MAC OS	Message Console
Linux	Kernal Messages

Back To Index

Biography

Joe DeMeyer has been working in IT for over 10 years in both development and testing roles.

He enjoys sharing his experience with his peers and the FIRST Lego League team he coaches.

To maintain his skills, Joe designs, builds, and, most importantly, tests small payloads and flies them using a weather balloon.

Contact Joe at Dev_To_Test@yahoo.com.

What is Application Transparency?

Testing requires a method to act on an application operation such that it produces a result. In many complex systems, the test result is difficult to observe or measure.

Application Transparency

- Provides visibility to application behavior and test results by having the application be more introspective
- Is a catalyst for test strategy development
- Facilitates conversations with developers to improve testability of their designs and programs

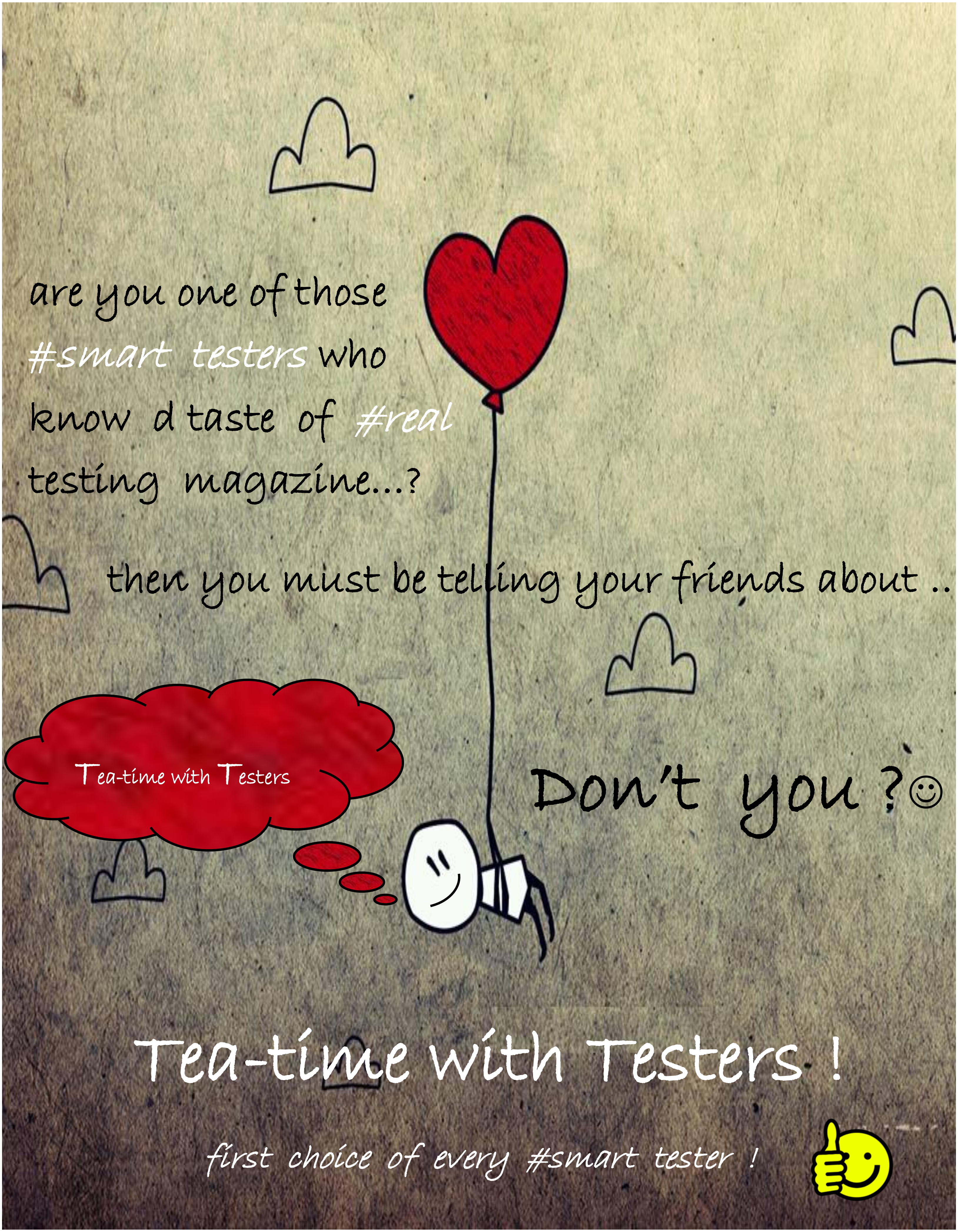
Application Transparency in Action

I have used these methods to help bring transparency to applications I have developed. In a testing role, I believe it enhances a structural test by exposing information not easily available by other methods.

I have proposed the introduction of transparency in two different applications. In the first, I proposed a stored procedure to insert records into a database. A process used to monitor the database for new records took action when the insertion was detected. The proposal was rejected. I suggested the stored procedure could go into production with no impact on the application. The team focused on this statement (rather than benefits of the test) and did not want to risk having a "test" stored procedure in production.

The second time I proposed a web service to return information about the machine where it resided. The web service was basically a smoke test. This, also, could go into production. While the team agreed with the test's merits, they felt information could be collected without creating a web service.

As a tester, I was not disappointed in either outcome. I felt I learned more about what project teams want to risk, learned that a transparency proposal has to have demonstrable value, and I have opened discussions on transparency for applications.



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

then you must be telling your friends about ..

Don't you ?☺

Tea-time with Testers !

first choice of every #smart tester !

testing intelligence

- its all about becoming an intelligent tester

an exclusive series by **Joel Montvelisky**

One metric = One Big Mistake

Sometimes measuring only one thing
can be worst than not measuring anything at all.

A QA colleague forwarded me a report he sent to his management where he pointed at the fact that the whole R&D Organization was not delivering products at the pace they used to do it in the past.

He was airing out a "known" general concern shared by many testers and developers, and to do this he searched for some "hard data" that would prove this point.

The report was very simple, and it compared the number of releases by all the teams for the last period vs. the period immediately before this one; and it showed that in the last period all the teams together had only delivered about 40% of the releases they did previously.

Simple, right? – **WRONG!**

What happened with the report?

Immediately after he sent the report, the Team Leads of two of the development teams answered with short emails "explaining" that in the last period they had also had a number of holidays, and attributing everything to this fact.

With this *excuse* the value the report went down and the subject was erased from Management's mind and agenda…

The "funny thing" is that my friend had also seen this point, but he thought everyone would see the difference was too large in order to be caused solely because of the holidays!

What can we learn?

There are some basic lessons we can learn from this issue that apply to most organizations

1. Don't use only one number in order to draw a complete picture

A basic mistake made by many testers and QA Managers is to base all their conclusions on only one number or metric. Basing everything in one measurement is like reaching your conclusions based on only one side of the story instead of collecting information from all possible sides.

So what do you need to do? Look for more than one angle in order to draw a complete picture.

In the case above, the manager could have looked for information about the number of issues detected during the process, or amount of rejections from the field, or at least start by "normalizing" the number of deliveries based on the total number of working days in both periods.

2. Take into account your workplace politics

It is naive to thing that you will send a report pointing at a serious problem and whoever is responsible for it will simply step forward and happily take the responsibility. V*ictories have thousands of parents, while defeats are usually orphans*.

This means that whenever you point to an issue in the system you need to make sure to cover all your angles and to provide information that is, as much as possible, irrefutable.

An important (& ethical) thing to do is to contact the person who may be responsible for the issue and make sure you communicate your findings to him ahead of time, before sending out the report to everyone else. This will give him a chance to come up with ideas on what caused this and even provide as some solutions that you can include in your report.

3. Metrics should include comparable historical information

In the case above, what could have taken down the argument about the holidays? To compare the number of deliverables not to the period before, but to the same period last year!

Not all metrics are trivial by themselves, and so you may need to compare information in order to see the real differences (or similarities) between them.

In the case of time-based measurements, you need to think about seasonality factors such as:

- Season or time of year
- Part of the quarter
- Time of day etc.

4. Take into account external factors

As much as you want to be able to measure everything with raw numbers there will always be things you cannot measure that affect your process. You need to research these factors and take them into account in your report.

In the case above, it would have been as simple as stating that in the last period there were also some out of the ordinary holidays, and then add a note saying that even if you factor them into account they would not lower the productivity bellow 80% of the regular level, leaving another 40% decrease in productivity unaccounted for.

External factors will vary all the time, they are the out-of-the-ordinary things that will be trivially visible for people who are living the process. Some examples can be:

- Extraordinary team growth or reduction
- Extreme product shifts or market disruptions
- Things like prolonged strikes
- And even in some cases, as I saw not too long ago in a company, unusual number of births in a team (5 out of 8 team members received a child in a period of 3 months)

Bottom line

One of the main difference between a credible story and plain gossip is that a story has more than angle to it. Your metric's-based-report is basically a way to tell a story of what is happening in your product/process/team.

So, in your next report or fact-based email make sure you take into account:

1. To have more than one metric or angle measured
2. Company politics
3. Historical information and
4. Extraordinary factor

Back To Index

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

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

Today Joel is the Solution and Methodology Architect at PractiTest, a new Lightweight Enterprise Test Management Platform.

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

Joel publishes a blog under - http://qablog.practitest.com and regularly tweets as joelmonte

Call for Articles !

Have you got something to say?

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

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

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

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

Click HERE to read our Article Submission FAQs !

sciencephotogallery

Taming Complexity

In my meetings with senior management and technical folks in high technology companies, the one phrase that I frequently hear is "My system is complex". When anyone utters this phrase I smile inwardly.

The way I interpret this is

(a) You do not understand the system or
(b) You are trying to impress me.

Complexity is an enemy and friend. An enemy because it can faze you, freeze your thoughts and numb you. A friend, because it depicts richness, the ability to do very many things. The trick is to be rich (in features/functionality) but not be fazed by it. Philosophical eh?

We all have intuitive understanding that complex systems demand 'more' or thorough testing. More testing is not about doing it frequently, it is really breaking down the complexity and ensuring that these aspects are well covered. We know that with experience, we handle this well as we are good at decomposing the problem and solving it.

Let's probe as to how we possibly do this. Let's start from the beginning. **"What does it mean when we say the system is complex?"** It simply means that there are numerous possible behaviors. Depending on the value(s) of the inputs the behaviors invoked are different and therefore generates different results. So, one of the aspects that govern complexity is the various conditions that govern the behavior. So if the entity's business logic i.e. behavior has quite a few conditions, then there are very many paths i.e. scenarios to test. To do this well, it is necessary to model the behavior using a suitable technique so that the complexity of the numerous conditions do not faze us. This is "Behavioral complexity".

The second aspect of complexity is in "how well the behavior is". What does this mean? It is about the expectation of certain attributes that the end user may be expecting when displaying the behavior, like speed/performance, security etc. In some cases the expectations on the attributes may be very high and therefore requires architecture/code to satisfy the demanding attributes. This is "Attribute complexity".

Let's move inwards to see the role of the internal structure towards complexity. The number of elements (i.e. moving parts), that make up the system, their interconnections and the way they are interconnected is what contributes to "structural complexity". Structure at the highest level is the deployment architecture, while it is code structure at the lowest level and the system architecture at the middle level. More moving parts, dependencies, interconnections at any of these levels imply that the system is "structurally complex".

Structure is not simply about code aspects, it is also about the structure of data. In some systems, the underlying data models can be involved and interesting and therefore we have "Data complexity".

The final aspect of complexity emanates from the environment- the number of environments that the system has to support, the dependencies the system has on the environment (e.g. setup, configuration...) and the rules of the environment that the system has to comply with. This is Environment complexity".

Let us a do quick recap now - we have broken down complexity of a system into five aspects

(1) Behavioral complexity i.e. the number of conditions that govern the system behavior
(2) Attribute complexity i.e. "how well the behavior is"
(3) Structural complexity i.e. #parts in the system, #interconnections and the method of interconnections.
(4) Environment complexity i.e. environments, configurations, rules of the environment
(5) Data complexity i.e. the inter-relations between the data (data model).

So when somebody says a system s complex, decompose into these five aspects and understand and rank these. This will allow you to get a handle on complexity and stay on top it, not be crushed by it!

Good testing is about understand the "complexity contributors" and pay extra attention by application of suitable techniques, expending the necessary effort and ensuring a risk-driven approach to validation to maximize business results. Hypothesis Based Testing (HBT) decomposes the system into five elemental aspects (Data, Behavior, Structure, Behavior, Usage) and one of the one core concepts "Complexity analysis" is useful in taming complexity. Look for http://slidesha.re/qBMNiy if you are curious about HBT.

So the next time you tell someone "my system is complex" Think! If you understand your system well, then you will indeed decompose it and describe it in simple terms. Albert Einstein said "If you can't explain something to a six-year old, you probably don't understand it". Hmmm. Imagine explaining General Theory of Relativity to a six year old!

Systems or life is indeed complex i.e. rich. Simplify it and you see beauty. The utter simplicity that hides the raw brutal complexity is what makes an end-user fall in love with the system. Well I am an Apple fan!

As a tester I enjoy grappling complexity because I am confident of taming the beast.

The second law of thermodynamics (Entropy) is alive and kicking in software systems too! Enjoy the richness of complexity and the beauty in making it simple.

Let your new year be filled with richness laced with the beauty of simplicity. Happy New Year.

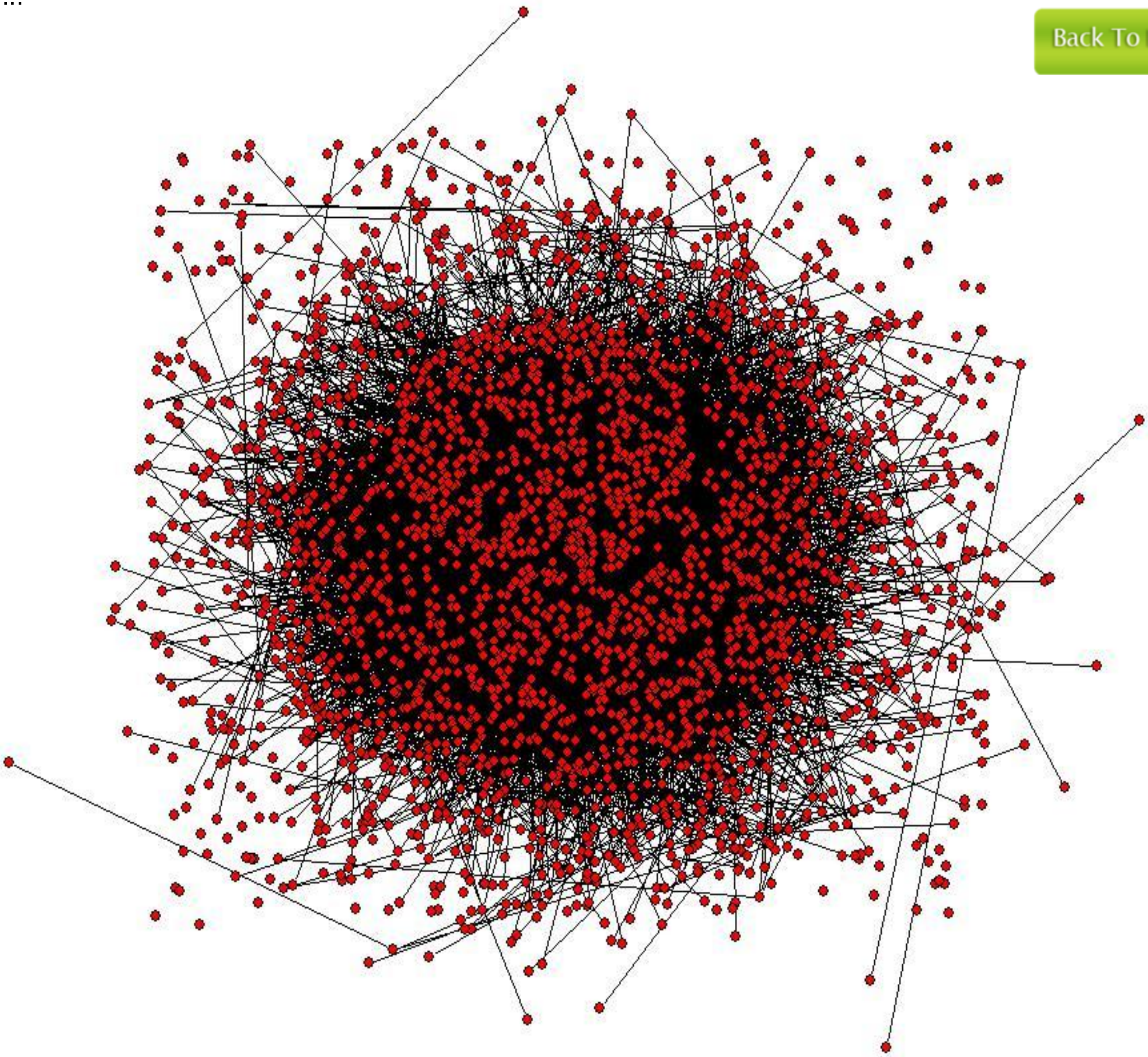
CIAO…!

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

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

He can be reached at **ash@stagsoftware.com** .

Back To Index



Season's Greetings to our all Readers

&

Happy New Year !

OUR PARTNERS

Quality Testing

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

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

Mobile QA Zone

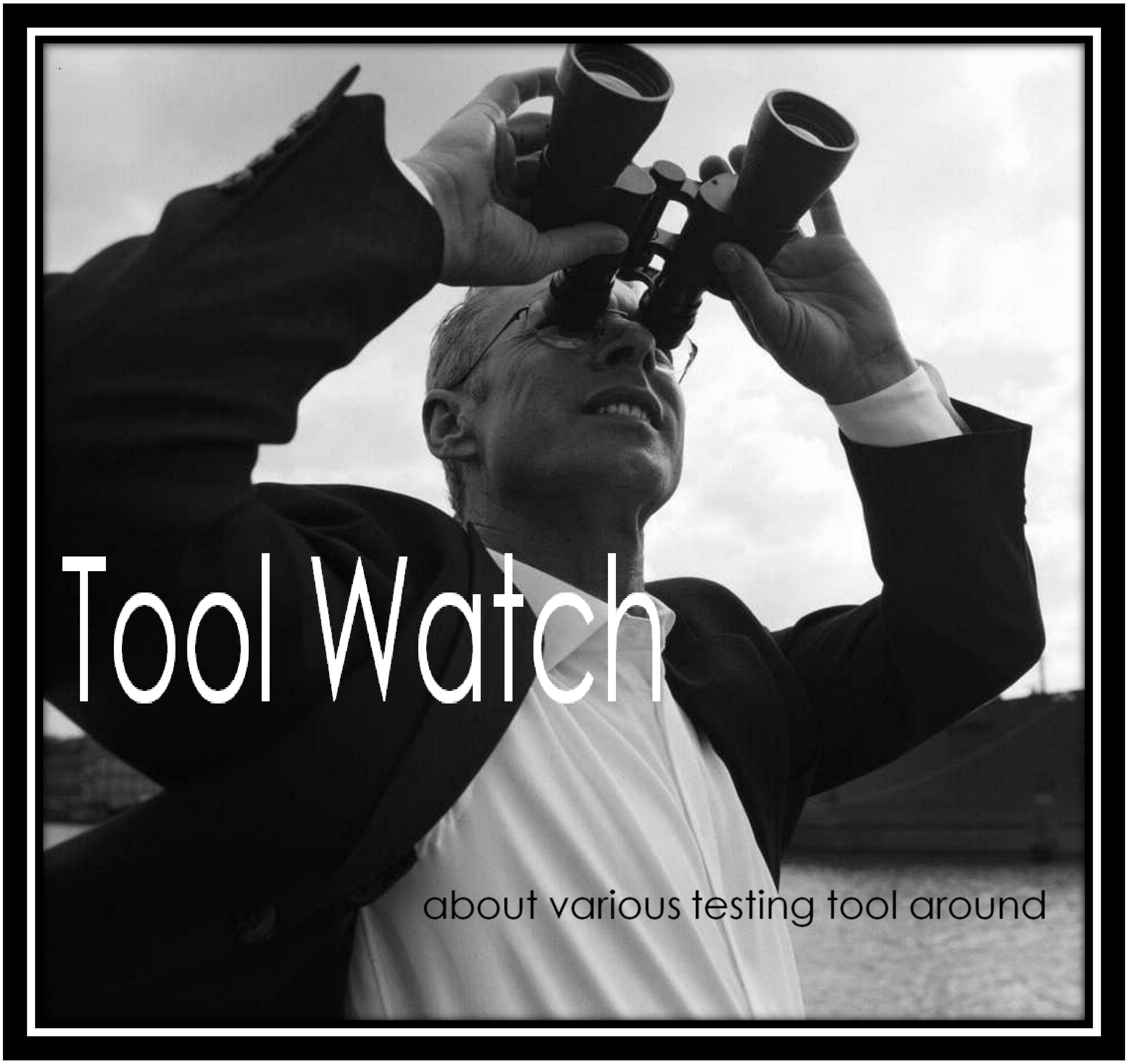
Mobile QA Zone is a first professional Network exclusively for Mobile and Tablets apps testing.

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

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

Tool Watch

about various testing tool around



Rapid Reporter

PART 1

By Chris Philip

Introduction

Rapid Reporter is a tool built in order to enhance the experience of Managing Tests Based on Sessions.

It aims on boosting good note taking during test sessions and building reports that invite analysis and discussion. Notes are easy to record and easy to debrief.

As a tester, you will find that Rapid Reporter will help you in master the session, doesn't interrupt your lines of thought and helps you keep track of notes during tests.

It may help you to perform the test better. If you are a test lead or manager or a peer tester, you will find that Rapid Reporter provides easy-to-study reports, which can be viewed and manipulated in a spreadsheet at will or viewed from a web-browser. This application was useful where the other options failed, due to being minimalist.

As it doesn't do much, it doesn't divert the tester's attention from the testing session.

Rapid Reporter is a work in progress, based on practical and ongoing experience, and has been evolving like that for a year.

Some of the benefits of Rapid Reporter:

- One single standalone file – Requires no installation, keep it in your disk-on-key and use it in any computer (that complies with the System Requirements).
- Yellow always-on-top interface – No more switching windows or looking for the note taking application.
- Every note has a type, a timestamp, and (optionally) two types of attachments: a screenshot, or an extended note (rich text).

- There is no editing or tinkering with the notes on-session, this can be done easily in or before the session debrief. This allows the tester to tell the story of his session in the order and with the details the session happened.
- The whole session, with reference to attachments, is recorded in a CSV file which provides easy manipulation (can be opened/filtered in Excel) and search indexing (find text in files). These CSV files can be consolidated or transformed into HTML documents with command line operations. A progress bar shows the tester how close the end of the session is. There are no noisy alarms, testers own their session time.

The following are the basic usage areas of Rapid Reporter:

1. Testing Notes

The first thing a tester does when firing Rapid Reporter is to enter two special notes:

His name ("Reporter:") and his exploration charter or mission ("Charter:").

After these two are entered, Rapid Reporter starts advancing the session timer (default is 90 minutes). The session does not really ends when the timer is over. The timer is only an indication, and testers can decide to stop before or after time is up. To keep notes, tester can enter notes in the notes line, and change the type by pressing the up or down arrow (even if he's on the middle of the note). Each note entered is saved by pressing the enter key.

There are two types of attachments: Screenshots and Extended notes.

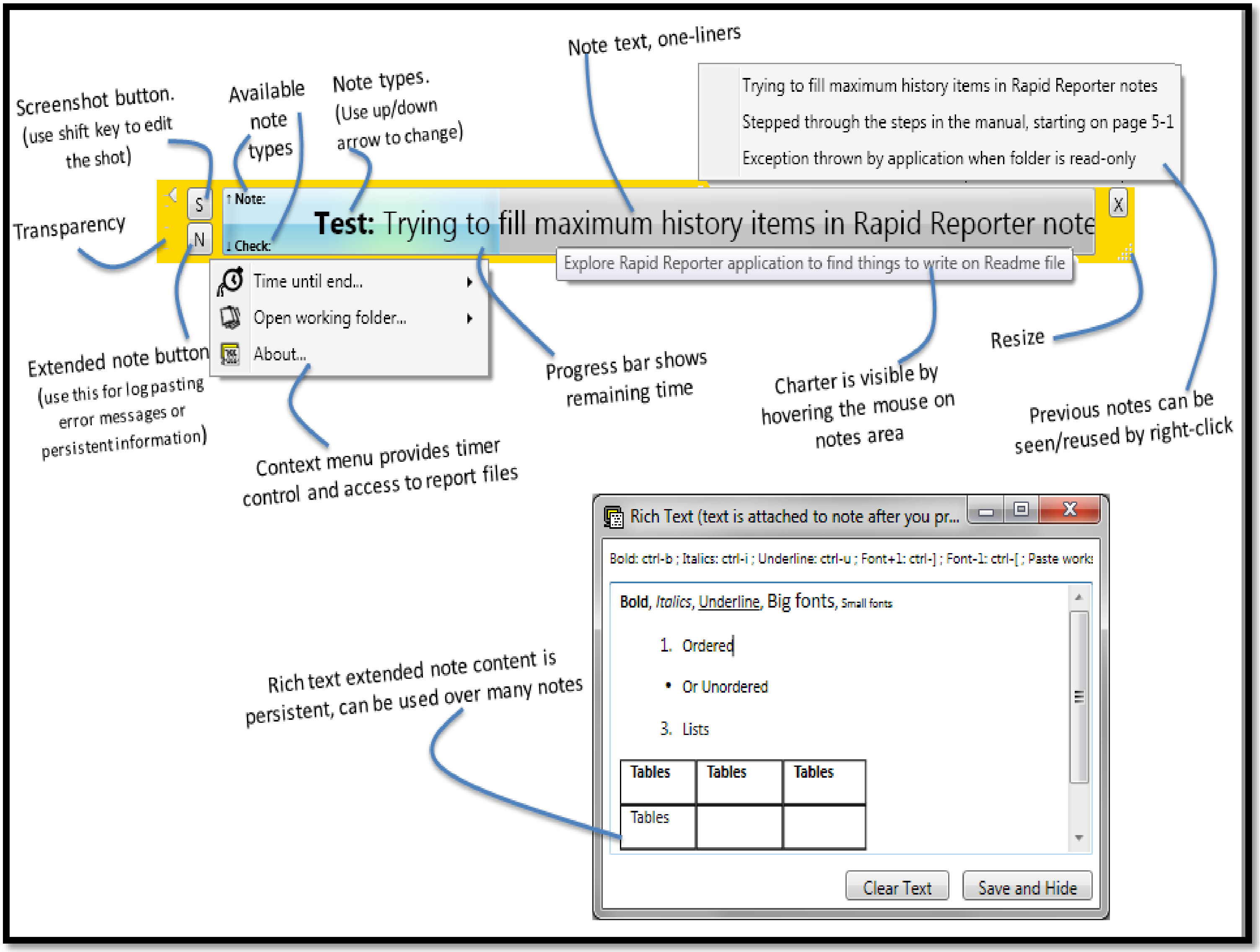
Screenshots are taken, saved and attached immediately when the screenshot button is pressed, without any intervention from the tester (but a tester can edit the shot if desired, by pressing the 'SHIFT' key while clicking the screenshot button).

Extended notes are also saved + attached at the press of a button , and have triple use: Adding more information to a one-line note, pasting test results or data to augment a one-line note, and keep track of information over time (notes entered in the extended text area are persistent until a tester clears them).

2. Visual Guide

Many prefer learning with visual clues.

Here's a summary (see the image on next page) of the different areas in Rapid Reporter's graphic user interface:


S
N
↑ Note:
↓ Check:
X
Time until end...
Open working folder...
About...
Rich Text (text is attached to note after you pr...
Bold: ctrl-b ; Italics: ctrl-i ; Underline: ctrl-u ; Font+1: ctrl-] ; Font-1: ctrl-[; Paste work:
Clear Text
Save and Hide

Trying to fill maximum history items in Rapid Reporter notes

Stepped through the steps in the manual, starting on page 5-1

Exception thrown by application when folder is read-only

Test: Trying to fill maximum history items in Rapid Reporter note

Explore Rapid Reporter application to find things to write on Readme file

Bold, *Italics*, Underline, Big fonts, Small fonts

1. Ordered
- Or Unordered
3. Lists

Tables	Tables	Tables
Tables		

Screenshot button. (use shift key to edit the shot)

Available note types

Note types. (Use up/down arrow to change)

Note text, one-liners

Transparency

Extended note button (use this for log pasting error messages or persistent information)

Context menu provides timer control and access to report files

Progress bar shows remaining time

Charter is visible by hovering the mouse on notes area

Resize

Previous notes can be seen/reused by right-click

Rich text extended note content is persistent, can be used over many notes

3. Session Debriefs

After the application is closed the session notes are ready to be reviewed. You will find all the files generated during the sessions at the session folder: *.CSV, *.JPG, *.RTF, *.HTM.

The session folder will be the folder from which Rapid Reporter was executed, unless you pick a different folder.

to be continued in next issue…

Back To Index

Do you think that even your own tool should be part of this unique section?*

Feel free to write us. Let the world know what your Tool can do !

To know more write to us at teatimewithtesters@gmail.com

* Conditions Apply

Chris Philip has been working with his first employer TCS since 2 years, passionately in area of Automation Testing. The passion and interest in automation was revealed during his college projects in robotics and successful completion of project work from India Space Research Organization, automating the microscopic and sensitive calculations regarding the minute changes in accelerometer data in launch vehicles, rockets and spacecrafts. The project was done in microprocessor programming language.

Chris is active member of Linux club, IEEE and Computer Society of India (CSI).

His special interests are software automation, having extensive hands-on experience in QTP, Sahi, Selenium and RFT.

Actively participates in blogs and discussions related to automation practices. He has presented 3 white papers till date about the automation practices, short cuts and interesting logics applicable in Quick Test Professional.

Chris is reachable at **christhomsonphilip@gmail.com** & on twitter **@chris_cruizer**

by

Quality is delighting customers
QUALITY TESTING

Testing PUZZLES

by Sebi

Claim your Smart Tester of The Month Award. Send us an answer for the Puzzle and Crossword bellow b4 10th Jan 2012 & grab your Title.

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

Gear up guys.......

It's Time To Tease your Testing Bone

Puzzle “Alphabets and Digits”

Replace the letters here with digits :

Two+Thousand-(and*twelve)=2012

Example : I*am+here=2*34+5676=5744 [I=2 a=3 m=4 h=5 e=6 r=7]

Biography

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

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

He tweets as @testalways.

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

Back To Index

TESTING CROSSWORD

1		2		3		4			5
■	■		■		■		■	■	
6				7		8	■	■	
	■		■		■	9			
	■		■	■	■		■	■	
10				11		12		■	
	■		■		■		■	■	
13	14				■	15			
■		■	■		■		■	■	■
16				17					

Horizontal:

1. It is a tool provide easy cross browser testing with Selenium in the cloud (10)

6. A test suite that exercises the full functionality of a product but does not test features in detail (7)

9. A test case design technique for a component in which test cases are designed to execute branch outcomes is known as______________testing? (3)

10. The consequence of a test (6)

12. Variation of regression testing is called _________ testing? (2)

13. It is a Java program that tests automatically and at random programs is called ______? (4)

15. The short form of Selenium Java Evidence (3)

16. Testing of individual software components is called______ testing? (4)

17. It is a tool provides automated functional GUI testing for Java and HTML applications (6)

Vertical:

2. It is a mobile test automation tool for Android, iPhone, Blackberry, Symbian & WindowsPhone7 (7)

3. Testing whether the software installation being tested meets predefined installation requirements, in short form (2)

4. An intense round of testing, quite often redirecting all available resources to the activity, in short form (2)

5. Data that exists before a test is executed, and that affects or is affected by the component or system under test is called _____? (8)

6. Ajax test runner for php, the first word (6)

7. Testing which demonstrates that the system under test does not work, in short form (2)

8. A test environment comprised of stubs and drivers needed to conduct a test (7)

11. The short form of Linear Code Sequence And Jump (5)

14. It is an quick and easy command line automation tool, in short form (2)

Answers for Last Month’s Crossword:

J	A	N	O	V	A	L	P	H	A	
I		I			D		F			
R	O	S	E	X	H	A	U	S	T	I
A		H			O					S
		A			C	U	B	I	C	O
C	A	N	O	O			R			L
O		T			A		A			A
D	E	B	U	G	G	I	N	G	S	T
E		T			I		C			I
		R			L		H			O
	I	N	S	P	E	C	T	I	O	N

Answers for Testing Puzzle of last month:

The longest string that is used as a sub-domain for a website:

http://zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzz.zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.eu/zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.php

We appreciate that you

“LIKE” US !

Join us on Facebook.

You are just a CLICK AWAY

Every Tester
who reads Tea-time with Testers,
Recommends it to friends and colleagues .
What About You ?

Image : vernhart

Our Testimonials

Hi,

I am a fresher student. And I have done diploma in software testing. I think this magazine will help me in gaining information about software testing.

First time I realized that there is world beyond text books and certifications and it is Tea-time with Testers which made me believe it.

- Vishal Kadam

www.TeaTimewithTesters.com ROCKS and is Awesome!

- Arun Francis DS

Well , interesting YES !

– Peter Vikstorm

Fabulous job by Lalit along with his team-mates. Today first time I read this online magazine.

All the topics are really appreciable.

It's really very creative, informative and interesting.

- Atulya Krishna Mishra

Great Magazine. I am still to finish reading the Nov 2011 edition, but it looks promising !

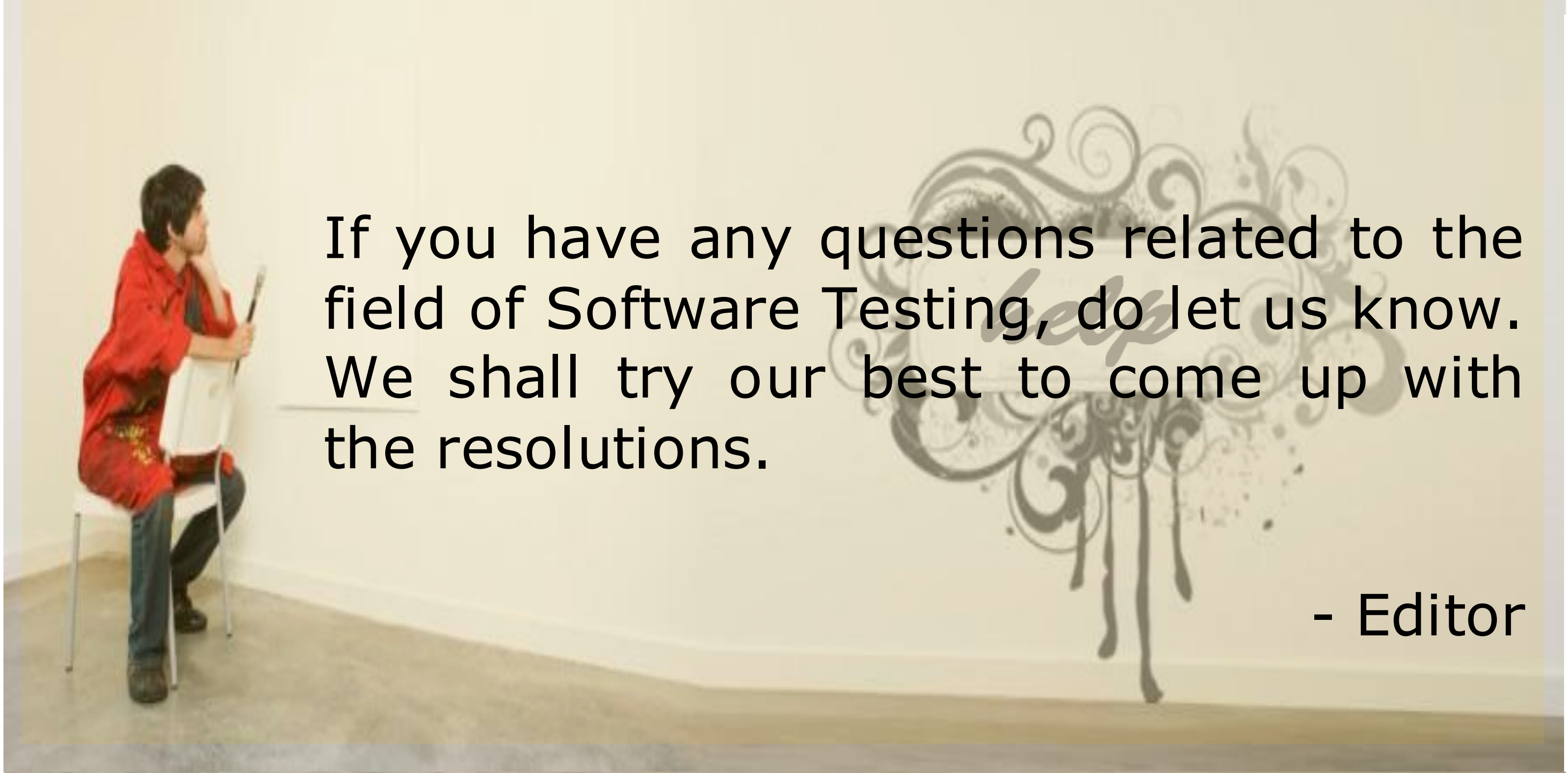
- Sultana Rassoul

First time I read it and I read it again. And am reading all issues since then. Thanks a lot for your remarkable efforts. Good Luck !

- Joshua Philip

You ask...

We'll help...!



If you have any questions related to the field of Software Testing, do let us know. We shall try our best to come up with the resolutions.

- Editor

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

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

in next issue

articles by -

Jerry Weinberg

T Ashok

Joel Montvelisky

Anurag Khode

Karen Johnson

Bernice Ruhland

Samarjeet Mohanti

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)

Lalitkumar

Pratikkumar

Contribution and Guidance:

Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)

Jerry

T Ashok

Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Cover Page Image- Roses and Teacups

Core Team:

Kavitha Deepak (Bristol, United Kingdom)
Debjani Roy (Didcot, United Kingdom)
Anurag Khode (Nagpur, India)

Anurag

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