

AN INTERNATIONAL JOURNAL FOR NEXT GENERATION TESTERS

Tea-time with Testers

APRIL 2020

AN AUTHOR'S VIEW OF TESTING

ISTQB, FEVER DREAMS AND TESTING

DEAR TESTING CAREER



HOW TO READ A DIFFICULT BOOK

CONTEXT DRIVEN TESTING FOR COMPLEX IOT

DISSECTING THE HUMAN/MACHINE TEST
CONUNDRUM



TEA-TIME WITH TESTERS

Created and Published by:

Tea-time with Testers.
Schloßstraße 41, Tremsbüttel

22967, Germany

Editorial and Advertising Enquiries:

- editor@teatimewithtesters.com
- sales@teatimewithtesters.com

Lalit: (+49) 15227220745

This ezine is edited, designed and published by **Tea-time with Testers**. No part of this magazine may be reproduced, transmitted, distributed or copied without prior written permission of original authors of respective articles.

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

Editorial

Between the Years

Believe me or not, but as I am writing this, my feelings are pretty much similar to how I felt when I wrote an editorial for the very first issue of Tea-time with Testers. We published several issues over the years and we indeed felt accomplished for the kind of contribution we made in the global testing community thereby.

Eventually, we stopped. For many reasons. The feeling of “we fulfilled our purpose” being one of them.

After the gap of some years in between, we are starting again. And the reasons are obvious. We see the need to once again do what we did years ago. To bring testers together, to give voice to thinking testers so they are heard in the chaos, to wake testers up and bring them out of their comfort zone, to prepare them to embrace the change, and also to help them stay firm on what must not change just to fit in. And the good news is, we now have Klára Jánová on the editorial team who will actively support me in doing that all. Welcome aboard, Klára!

Between these years, certain things have changed a lot and a lot has not changed that much. Despite my best efforts to see good in it, ISTQB continues to disappoint me (and people like me), clearly pushing the ideas that suit its narrative. The magic wand of automation continues to sell and organizations continue to clap for that magic, without realizing how they are getting tricked in the hindsight. QA /Testing continues to be a bottle-neck for many who call themselves Agile but are still unaware that quality is a team sport. Sigh!

On the good side of things though, I see testers going beyond their traditional role, becoming more tech-savvy, acquiring and using the programming skills for doing meaningful automation in testing, taking responsibilities beyond their traditional role and helping teams beyond testing tasks. Those who got it, are helping teams build quality products using testing education as a tool to facilitate it.

On contrary to a popular manufactured and marketed belief, I firmly believe that testing is becoming more human than ever. No doubt that tools have evolved in many ways, but automation in testing has existed ever since. The direction testing is really heading to nowadays, as I see it, is more towards humans who have hardly known testing in a meaningful way. Understanding different notions of software quality and using testing education as a tool to facilitate that learning, is becoming increasingly important across all roles within the software development sphere.

To build software by humans that are meant to be used by humans, testing better be significantly human. And it better be part of some human’s responsibility, even better if of all who are part of building the software.

Until next time, fellas!

– **Lalitkumar Bhamare**

editor@teatimewithtesters.com

@Lalitbhamare / @TtimewidTesters



QUICK LOOK

SPEAKING TESTER'S MIND

- 04 *AN AUTHOR'S VIEW OF TESTING*
- 09 *ISTQB, FEVER DREAMS AND TESTING*
- 17 *DEAR TESTING CAREER*

IN THE SCHOOL OF TESTING

- 26 *HOW TO READ A DIFFICULT BOOK*
- 35 *CONTEXT-DRIVEN APPROACH FOR TESTING COMPLEX IOT SYSTEMS AND DEVICES*

T' TALKS

- 46 *DISSECTING THE HUMAN/MACHINE TEST CONUNDRUM*

QA INTELLIGENCE

- 49 *SELF-CRITICISM, THE KEY FOR CONTINUOUS IMPROVEMENT*



SPOTLIGHT

ISTQB, FEVER DREAMS AND TESTING

What is the future of Testing for ISTQB and rest of us? /p09



DATASET CHALLENGE

An Algorithm for Stories
on Women in History

#VoicesofWisdom



IVOW

TALK OF THE TOWN

TESTERS, STORYTELLERS AND CULTURAL AI

Testers engage in future of cultural intelligence in AI /p51



A photograph of a green, conical weight hanging from a thin string. The weight is positioned over a circular pattern drawn in the sand. The sand is light-colored and shows some texture and small debris. The entire scene is framed by a dark blue border.

Speaking Tester's Mind

- straight from the author's desk

An Author's View of Testing

- Mike Lyles

In 2002, I decided that I wanted to keep a journal for my professional career. In the early pages of this journal, I wrote the following: "I want to write a book someday". I knew, even back then, that my ultimate goal was to write a book that would be inspiring and motivating to the world.

I began taking notes in my journal, logging ideas the moment they came to me. These ideas and notes were disorganized and coming from every direction. My goal at that time was not to finalize the whole book but to get all of my thoughts in one location so that I could refine them. I also realized that I wanted to publish a book that was not one story from front to back. Instead, I wanted my book to be multiple chapters, each with its own story and takeaways.

Through the years, I would think of a great chapter title and write that down in my journal. By 2014, I had almost 40 titles in my list, and I began to build the stories that would map to each of those titles. After 5 more years of work, planning, refining, and editing, my book was published in November 2019 in paperback and eBook format across the world.

When I look at my long career in IT, particularly in software testing, I realize so many similarities in testing and the process that I used to get to the final published book. Below are those similarities and how you can take away some lessons in testing from each:

Start Before You're Ready

The first chapter of my book (“Just Start It”), discusses this in detail. Too many times we wait too long before starting the important things. Many testers will wait because they want to have all of the requirements or project details before they get going. They want to have all of the pre-requisites in place before they begin the testing process. But this is not the best way.

When I started my book, I had no idea how the final copy would look. In fact, my final printed copy looks much different from what I started with. The success of my book came from pulling together everything I could think of and getting all of my thoughts together in one location so that I could move to the next level and refine the chapters to prepare for printing.

As testers, we can benefit from pulling together everything we can about the product. We can study how it works (or will work), review artifacts (if the product exists already), and learn a lot about the product and how it should work, with or without the standard preliminary documents and prerequisites. Learn as much as you can about the product and begin to put your ideas together on how you will test it. And do this as early as possible.

Learn from Others

They say that the best way to learn how to do something well is to listen and watch others that have been there before. With writing, I knew a lot of authors and writers that had published books. They had the war stories and lessons learned to share with me. I listened closely to their advice on the things I “must do” as well as the mistakes I should try to stay far away from making. In my discussions with each of the experienced writers, I learned many tricks on how to work with the community, how to gain support from other authors, receive endorsements, and how to market the book before and during the printing and publishing process.

As testers, we have such a strong community out there that have been there and have lessons learned for us. Get out there in the community in social media – on LinkedIn, Facebook, Twitter, blogs, webinars, and more. Learn from their inputs and suggestions. Talk with the leaders and collaborate with them to help you grow as a tester and build your skills.

Refine, Refine, Refine

When I started my book, I never realized that on the road to publishing, I would read my book nearly 10 times from front to back. When I was wrapping up my book and giving myself a high-five that I was “done”, I had no idea how “not done” I was at that time. I began to edit the book, and each time I would find something I wanted to change and modify. I then hired an editor to look at the book and give their thoughts. They came back with many changes – suggestions that were so good, but I had not seen them myself until the editor called them out. After the editor completed, I read through the whole book again, made more edits, took their suggestions, edited again, and then shared the updated manuscript with the editor once more. The second pass was better, but there were still findings and updates that needed to be made. After going through it many times, making updates, I finally reached the last version and was ready to submit to the publisher.

In testing, we should drive for the same process. Do not be satisfied with your first test artifacts. Do not just settle when you feel you have put it all together. Give your artifacts a second review and look for areas of improvement. Reach out to others on your team to get their inputs and suggestions. Having that second set of eyes on your work gives you an opportunity to receive suggestions that will improve your work. As you progress with an application or system, you will find that things that worked long ago may need to be improved so that you continue to give the best testing experience to the deliverable. Keep refining and improving.

Cut Out the Waste

One of the most difficult feedback to hear in my writing journey was from a 12-time best-selling author and friend. His suggestion to me was to take a hard look at my manuscript (which, at the time, was 45,000 words) and trim the book down to a digestible and better product. He called it 'brutal editing'. He reviewed my manuscript, took a few chapters, and gave me examples of how to cut and trim the chapter to a better product. It was hard to do. I felt I had poured my heart and soul into the chapters and at first, I wanted to reply "But this is my book baby! Every word is important, how can I cut any of it out"? But I found that he was right. As I read through the document, I found better ways to say the same thing with much fewer words. It made the manuscript easier to follow, and it got to the point with minimal rambling. At the end of my brutal editing, I had a manuscript that was around 34,000 words and 30 chapters. And it was such a better product.

As testers, we hold on to old processes. We may be required to write scripts, steps, scenarios, create tons of documents. And we may or may not agree with everything we are doing. Our goal is to refine our processes and documents. We learn as we go, and the more experienced we get, the better we are at how we document our processes and results. Don't fall into the trap of feeling that you have to build long detailed documents to show your plan for testing or your results. Get to the point and get there fast. Continue to do the excellent work and execution in test, but trim down on what is unwanted, less useful i.e. make it lean.

Find Your "Endorsers"

Each of us have opened a book to the first page and seen high praise for the author in those first pages. This praise comes from famous people, famous authors, well known people in the community, and then also people that you may not know so well. Either way, the quotes from those endorsers are always positive (of course they are, who would put quotes in their book from folks that did NOT support them positively?). These quotes give you an idea of how good this book will be. As you read the quotes, you begin to tell yourself "I can't wait to read this book. Look how much these people love it!"

Get yourself a mentor. Read this again...Get yourself a mentor. Everyone needs mentors. And you don't have to pick just one. You can have a mentor for your work and career. You can have a mentor for your personal life and goals. You can have a mentor for all aspects of your life. But learn from people that can help you and endorse you. Reach out to those folks that you know have been there, going through the tough times, and have learned from it. Get their support and endorsement in your work and career. As testers, find those people that collaborate with you and can teach you new and creative ways to do your job. And most of all, find those people that will be there to support and tell others about you when the time comes.

Share Your Story

The feeling of relief when my book first hit the stores was overwhelming. The feeling of satisfaction as I held my paperback in my hands was so wonderful. Seeing my book available at every online book store was fulfilling. But then the reality hit me. I now have to MARKET MY BOOK! It was not going to market itself. Of course, family and friends would buy my book, marketing or not. But to really get my book out there into the world would take a marketing plan and a strategy to get the word out about my book.

As testers, you will find that the people working with you may know the great job you do, and they may even know the tricks and trips that you use in your daily routine. But there are testers all over the world that need to hear your story. They need to hear your successes (AND your failures) so that we can learn together. Therefore, I suggest that you get involved in local testing meetups, testing conferences, and share your stories and lessons learned. There is someone somewhere waiting to hear what you have to say.

You may be saying “I’m not brave enough to stand at an event or conference and talk – I am NOT a speaker”. This is totally okay. If speaking scares you, then write blogs, articles, and other materials for the public. Interact with people on social media and join groups on LinkedIn, Facebook, and other sites. Sign up to give webinars (they can’t “See You” if you do a webinar). Regardless of how you get your word out, just begin to share your story and you’ll find there are many people you want to hear it.

I have enjoyed writing this article and comparing the similarities between testing and writing a book. I have found in my years of writing, speaking, and working in testing that you can relate testing to so many areas of life and so many comparisons to be made. It has been a pleasure to share this experience with you and I do hope you check out my motivational book, “The Drive-Thru Is Not Always Faster”, which is available on all book sites (e.g. Amazon, Barnes & Noble, Apple iBooks, BookBaby, and other sites where ebooks and paperbacks are sold). A quick link to all the sites can be found on my website at www.TheDriveThruBook.com

Mike Lyles is a Director of QA and Project Management with 25+ years of IT experience in multiple large organizations. He has held various leadership roles: software development, PMO, and software testing. He has led various teams within testing: Functional, environments, SCM, TDM, Performance, Automation, and Service Virtualization.

Mike has been successful in career development, team building, coaching, and mentoring of IT professionals. Mike has been an international keynote speaker at multiple conferences and events. Learn more about Mike & his book at www.MikeWLyles.com & www.TheDriveThruBook.com Mike’s Social Media:

Twitter: @MikeLyles

Instagram: @MikeLyles

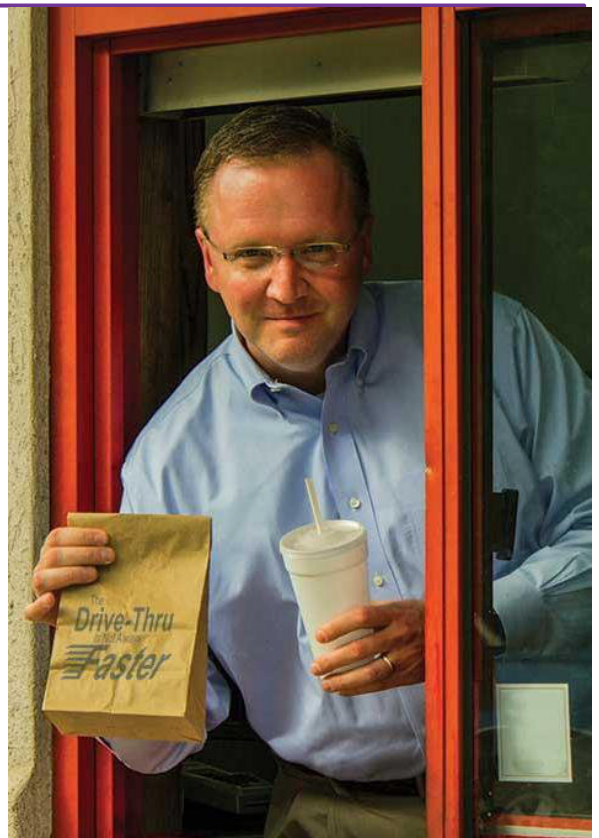
LinkedIn: <https://www.linkedin.com/in/mikewlyles>

Facebook: <https://www.facebook.com/mikelylesbusiness>

YouTube: <http://www.youtube.com/user/mikewlyles>

Website #1: www.MikeWLyles.com

Website #2: www.TheDriveThruBook.com



The
Drive-Thru
Is **NOT** Always
Faster

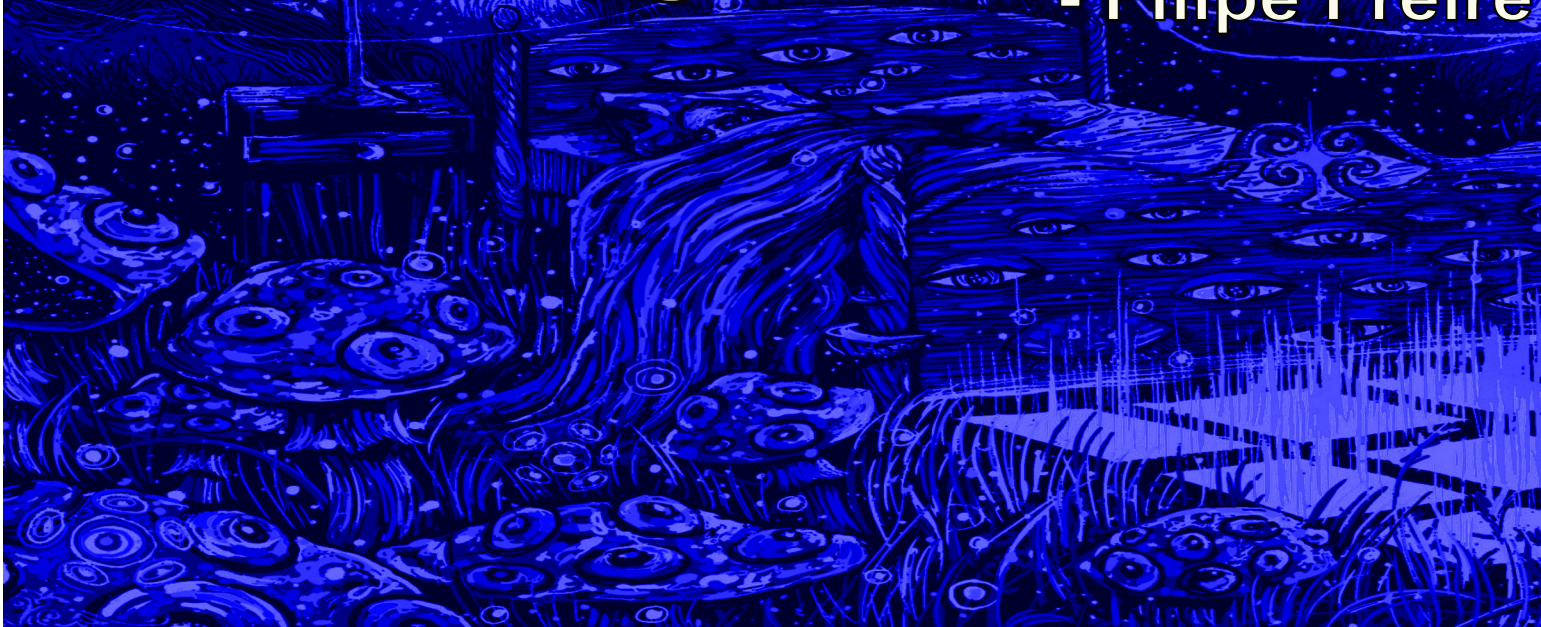
Inspiring stories from everyday life experiences

— M I K E W L Y L E S —

www.thedrivethrbook.com

ISTQB, Fever Dreams and Testing

- Filipe Freire



Picture by James R. Eads

Someone over at *self-proclaimed united nations of testers*, also known as *International Software Testing Qualifications Board (ISTQB)* published recently a letter (a white paper really) of what I assume is the result of having a fever dream after having watched *Blade Runner 2049*. Something tells me they probably were left thinking that the *Wallace Corporation* is the hero of the story (For those who've never watched that movie, a similar analogy is believing the iceberg was the lead hero of the *Titanic* movie).

Jokes aside, my belief is that what the white paper means for the industry and the testing professionals **should not go unchecked**. You can read the full thing [here](#).

Here's a few excerpts on the white paper:

"Software testing is moving towards more automation"; (...) *"In the future, there will be more testing technology solutions" (...)* *"The testing technology solutions will have 'intelligence' built into them." (...)* *"These testing technology solutions will iteratively learn the testing discipline from testers and testing experts" (...)* *"The testing technology solutions will also learn from themselves and, therefore, teach themselves as part of the iterative learning and evolving testing discipline, especially emphasizing testing quality attributes." (...)* *"The testing technology solutions will have the ability to explain to testers and testing experts the rationale supporting their recommendations and solutions." (...)* *"As more software solutions are being made available for use and as these software solutions expand into more industries, there needs to be a greater definition and integration of the training solutions for testing professionals." (...)* *"Testing professionals shall need to embrace the new methodologies and technologies around software development and testing, with intent to integrate these faster into their testing technology solutions." (...)*

If you know me personally or from my blog writings, you probably guess what I'm thinking: A lot of potential hogwash in so few sentences, to avoid calling it something else.

But I for one am going to hold my feelings for a moment, let's give the white-paper the benefit of the doubt and dissect a bit what this might mean, and I'll be doing it keeping in mind:

- **The people** who have respect and love for the Testing craft, unlike the ISTQB as an abstract group (and I'm willing to grant, not necessarily all of its individuals, since I know quite a few of them who love and respect Testing);
- **The people** that in one way or the other oppose the ISTQB angle in the craft;
- **The people** whose pockets and savings accounts are in the ISTQB crosshairs;
- **The people** whose pockets and savings accounts are in the crosshairs of groups or individuals that in one way or the other oppose the ISTQB angle in the craft for their own gain;
- **Anyone else** who may care.

There are three points I think makes sense to explore: "automation", "intelligence" and "embracing" the ideas behind the white-paper.

The "move towards automation"

The "saying" that the craft is pointed in the direction of automation holds some truth to it, as well as horse droppings.

It's undeniable that, from a "buzz-word" standpoint, indeed Software Testing seemingly has been moving more towards "Automation". The common industry accepted understanding is that there is such a thing as Manual Testing, and the paired industrial understanding that where you see the word "Manual", there's bound to be a machine that can do it better, faster, cheaper. And in this sense, there's no mute button for reality, and the reality is: New tools. New software libraries. AI advertising of "it'll do it for us". New ways of representing a view in an object. New ways to call an API or publish a message in the context of a scenario that will assert something in the end. New ways of organizing "testing" code. etc. Even personally, it's undeniable: I spend a portion of my working day coding, using tools, and maintaining scripts, and I am a Tester.

The horse dropping part of it all comes when you confront the "buzz-word" with different abstractions of the whole actual Testing phenomenon, and in its truest raw form:

Testing is undeniably human.

There's simply no way around it. Reducing Testing to logical assertions written in a programming language, or mixing that with some "intelligence" (aka. some "dumb" machine learning model), is indeed a faulty approximation of Testing as a whole, that comprises: investigating, exploring, purposeful play, dealing with confusion, risks, problem hunting, communicating, representing, and yes, also logical checking.

Excellent testing is a mix of all of these things, and all of them bear the undeniable mark of their creator: the human being.

The confrontation of the "buzz word is actually horse droppings" has been happening now for maybe almost two decades (maybe more), and folks left and right have come forth on plenty of occasions saying both "true-ish and horse dropping" words of disorder: "Test is Dead", "Automation Testing is the future", "Bots will replace human testers", "Agile testing is a thing", "The death of manual testing is NOW", "Let's all get along, testers are cheerleaders", "The role of QA in Agile Teams" ...

The above ends up being paced alongside their counterpart: individuals and small groups of people pointing out that:

- No, testing isn't dead, it's more alive than ever. In fact there's an ever growing hunger for testing that is not braindead but is rich with substance and purpose;
- No, you can't automate testing, because testing is not only "logical checking";
- No, bots can't replace humans in Testing. They can aid in specific tasks;
- No, agile testing is not really a thing. You have testing and its surrounding context where it's inserted;
- No, manual testing is not really a thing, the same way it doesn't make sense to say manual thinking, manual questioning, ...;
- No, we can't get along. We can still be friends outside of work, even lovers, but we need to have some level of critical distance to hunt down certain bugs and problems;
- No, I can't QA that for you. You want me to TEST it? Did anyone tell you yet the [meaning of Quality Assurance?](#)

And there are always these two sides, who both offend and feel offended, invested in a struggle between

- those who have over-simplifying models for something chaotic that is strongly dependent on its context,
- and those who oftentimes enclose themselves in tribal behavior, whose ideas suffer for being produced by faulty and imperfect humans.

Curiously enough, I think both sides would be able come up with ways to tag any of the above descriptions to the opposing side. Every year there's a whole living and breathing market for trainings, conferences and posts built around promoting or deconstructing these words of disorder, propagated globally faster and faster on every iteration. *Such is life.*

The “intelligence”

I admit there's bound to be software now and in the future that can mimic certain aspects of our natural intelligence. I also admit there is the possibility to build intelligent agents who can greatly surpass humans in certain specific capabilities or very specific activities.

To be honest, also I'm not prepared to go into all the ethics and existential issues that surround many of these topics. I have zero desire to trash the past and ongoing work of people who are dedicating their lives towards investigation in artificial intelligence, machine learning, or its several brain-children like deep learning, and networks and specific algorithms.

But, I can talk about what I've experienced first-hand in one story to elaborate on my skepticism on any “intelligence” ~~sales-pitches~~ arguments.

My skepticism on implementation

There's a couple of startups spewing around selling the “idea” (and implementation) that they're developing “robots” with some capability of learning (in the shape of software as a service) that will go through your App or API and will signal inconsistencies, possible problems, and even predicts pain points, all in neat dashboards and well integrated with any typical continuous-integration pipeline.

As Mr. MacMahon would say in character in a WWE show if he were a software person - *“We're developing this solution to finally replace those filthy and smelly humans and- *GASP!* (he gets surprise whipped by Steve Austin)”*

If it's intended to help/assist, then I don't really have a problem with software that helps me, or conveys me some bits of useful information. Anyone who knows me knows I'm a somewhat of a freak adept of tooling like [\(hardcore\) static analysis](#), proper unit tests (aka checks), and using code quality tools in a hardcore and extensive way if we're serious about the whole deal, and specially in open-source projects.

The problem is this ~~crap~~ stuff is advertised as “the next step”, one to replace Testers in practice.

Added to the mix, it might be the case the main “gurus” behind the foundation of said startups and software have very established credentials, and a subliminal, if not natural, disdain for the Testing craft (oversimplifying it to plain checking at different levels) and in terms of their usual lingo are the typical shallow “automators” - meaning they've done in part the same I also do everyday - coding automated checks - but they elevated the checks to a replacement of deep and meaningful testing.

Now, I won't go ad hominem straight away. They don't love or respect the craft with their time given on this earth, that's fine by me. But using their *established status* or their history,

“I worked at EvilCorp for N years as Omega Director of Testing & Quality” (Yup, Mr. Robot reference)

to sell “air in a jar” software, directly mocking Testers, that I can't stand.

Opening the “air jar software”

I invest some of my free time hacking away at any piece of software I can get my hands on. One of those sessions last year was spent hacking away at one of these fancy “Automated AI Testing” services. After some time tinkering around I was able to notice it didn’t actually do anything as advertised, and their entire Admin API was badly protected, so any attacker was also able to create accounts with admin permissions, spam the service with crap data and do a couple more mischievous stuff. Suffice it to say:

- No, it didn’t seem to work as advertised, it didn’t replace me as a Tester, and it didn’t even replace the most clumsy open-source static analysis or code quality tool out there;
- If any of my past, current or future employers would ever upload a build artifact for that service, I would be the first one to warn that any script kiddies will be the first to get their hands at anything sitting in storage on that service.

As an extra point, and none of these companies will probably do this in the next 10 or more years, if there’s a service that “tests” my build artifacts, the only feasible way I see of remotely adopting this as a **helper** stage in my CI pipelines is if I have full open access to what exactly that service does, and the “if-else” chains in the scripts it’s made of (advertised as AI). And by open access, I mean also open-source access to its code. Why? Two reasons:

1. Evidence of testing & Test Reporting.

It’s as simple as this: as a Tester or a Test Manager I expect myself or any of my Testers to produce and convey evidence of their Testing. I want to understand at any stage in a clear and human way:

- what was tested?
- how was it tested?
- what was left out?
- which tooling and automated checks are in the scene?
- what was confusing?
- what wasn’t looked at?
- what are the main risks for different actors (example: what’s important for say, a regular coder, is not necessarily important for a sales rep, and vice versa...)?

And no, I’m not talking about some shallow test-case counts, percentages, graphs, push notifications, some “superficial” test reports, among many other things like this that are sold as “evidence of testing” or “quality reassuring”.

2. Power to the people. Yeah I’m going that way. Sure, GOOD Testing is expensive. But it is also my belief Testing is “free”, and I mean it as “free” as in “freedom” (not necessarily as in “free beer”). Knowledge surrounding testing and other topics should not be contained behind a corporate or academic veil. It should be free and available for the thirsty.

Building a monopoly and then charging money for infrastructure plus some obfuscated service that has scripts that perform some automated checks on your build artifact is as sketchy (if not more) as it is selling licenses (plus painted-over infrastructure) for “load testing tools”, when there’s plenty of open tools that work better in almost all cases, it’s just that there’s usually laziness involved in the load generation infrastructure setup part, and excuses start popping up.

Embrace or perish

If it’s by chance, it’s probably an unfortunate one that we read throughout the white-paper:

(...) there needs to be a greater definition and integration of the training solutions for testing professionals (...) (...) professionals shall need to embrace the new methodologies and technologies (...)

And let’s think for a moment... who are the self-proclaimed responsible entities that we know about that by mere causality... are the de-facto go-to “industry nice-guy/nice-gal” that sells, “validates” and “certifies” training solutions for testing professionals?

I’m not trying to imply anything, but let’s take this to an imaginary surreal fairy-tale scenario...

A tale of towel salesmanship, by Filipe F.:

I’m born Portuguese, so I’m by default a [certified Towels salesperson](#), right?

Let’s say I’m also the founder of International Jedi Council of Portuguese Towels (aka the IJCPT). We promote self programs and certifications to validate other well respected and certified Towels salespeople like myself.

So tomorrow I go about my business and announce to the world that the future for towels-selling is Towels that have a special fabric called “s-moke”. And where can people get training and certification? Well I’m happy to say that the IJCPT is keeping up with these latest industry trends, and fear not, we can train sell any decent and concerned Towels salesperson preparation.

There’s a slight problem with the IJCPT... you see, we have a whole organizational structure, a bit like one of those genuine-honest-to-god-not-a-pyramid-scheme multi-level marketing institutions that are around? You know those? But hey, you can trust me, I’m trusted in all things Towel-selling by the whole industry by all my non-Towel-selling colleagues (like Senior Leaders, Recruiters, Managers, ...).

And for the record, I’m not saying “s-moke” Towels will 100% certain be the future... but I predict they will, and you should all pay attention to what I say.

So don’t forget: In practice, in this industry, no one else is acceptable in the eyes of God or Nature to “validate”, “certify” and “assure” your Towel selling skills. So you better take my ~~expensive~~ affordable certifications and trainings or you’ll be out of a job once the “s-moke” Towel technologies reign.

Fin.

Not implying anything, but you get what I mean, how this all might look weird for any “layman” in the end?

A last remark

Enough ranting. I'd like to direct this part to the team behind the white paper, if they ever read this post:

Yaron Tsubery, Dave Miller, Patricia A. McQuaid, Ceren Sahin Gebizli, Lucjan Stapp, Dani Almog, Filipe Carlos, Miroslav Bureš, Ritendra Banerjee, Adam Roman, Capers Jones, Alan Page, Bill Helfley, Colin Onita, Erik van Veenendaal, Harry Sneed, Jakub Rosinski, Klaudia Dussa-Zieger, Mark Gillenson, Olivier Denoo, Padmanabhan Balasubramanian, Paul Xzhang, Przemyslaw Niedzialkowski, Radoslaw Smilgin, Robert Binder, Robin Poston, Stephan Schulz, Steve Hutchison, Yao Shi, Alon Linetzki, Armin Metzger, Jasbir Dhaliwal, Kari Kakkonen, Leanne Howard, Michael Stahl, Tamas Stoeckert, Tilo Linz.

Fellas, I'm just a random dude. Not accomplished in much. Definitely not a bright mind. Ask some of my family members or old and current colleagues, they can tell about all the times I was a jackass. It's not necessarily brave from my side to also just take on some "group", without considering there's real human beings behind it. You wanna make a buck selling (or imposing to the industry) training and "I'm certified" stickers, be my guest: I don't like it. It's no secret I'm against it, and my personal experience tells me time and time again, there is indeed a better way. But really, I'm not in an ideal position to judge any of you.

I would like for you, if any of you reads this, to keep this in your hearts: the future we will all get to live on the little time we have on this earth, in part is a future of our choosing, both individually and collectively. You've made your choice, and probably the near future will prove it a right choice for your pockets, personal well being and standing in the overall community. History is curious though, everyone says History doesn't forget.

"People don't forget, nothing gets forgiven" would a character say in RDR game.

We unconsciously post our inner lives online, fly on planes, install security devices in our homes, drive to work, order food online, check our savings in a half second, invest in Tesla stocks with a tap of our fingers, and later on we will get automatically transferred our retirement money every month. Each of these things and so many more of them are built by imperfect humans, tested by imperfect humans (when and if they ever get meaningfully and decently tested), and sometimes, sure enough, kill imperfect humans. And I'm afraid, because we don't have to go very far off into the future, the present times prove to us that to make an "intelligent automated software testing" soup, or "half-assed certified" testing efforts, and think it'll remotely replace a craftsman and imperfect tester, is an entry way for more and more embarrassing bugs, which will potentially lead to the demise of many fellow humans. And the first responsible wont be the tester or the developer who either missed finding the bug, or inadvertently coded the bug. Neither will it be the greedy business man standing behind them.

The first people responsible are the educators and in the limit, the "certifiers". You may never be convinced that Testing is human to its core and instead paint it over with terrible bad jokes of science fiction (unworthy of the less good P. Dick or R. Heinlein novels), but keep in mind there are always consequences, positive and negative, in educating about Testing, whatever your angle is.

As educators, never forget that if we live, breath and mostly make a profit of "defined and integrated training solutions and certifications", for our fellow imperfect humans, nothing will ever clear any blood we may gather on our hands through time.

[Take care.](#)



Filipe Freire is a Software Tester, and a Developer. Currently working with adidas being the Lead Internal Tester for adidas App team and the Lead Tester for a classified high-value project.

His main interests are software testing, testing web applications, web APIs and mobile applications, performance testing, bot hunting, working with OOP languages, developing web applications and contributing to Open-source Software.

Visit his blog - <https://filfreire.com/>



ConTEST
NYC 2020

November 16-20

Call for proposals >

Dear Testing Career



- Karen Nicole Johnson

Dear Career,

This is a letter of farewell and thanks to you, my Career. Yes, that's right – my 33-year career in computer software testing is coming to a close. I am making this choice after heavy thinking throughout a sabbatical that has spanned many months and capped a change that had been simmering for the past five years.

Writing this farewell to my Career has become important to me. Important to publicly announce and acknowledge this change as opposed to withering away or disappearing. Important to me as a reflective activity that is helping me prepare for my next career. And, in tech (and perhaps as most industries these days), no one will be giving me a gold watch to say thanks and to send me on my way, so I am doing this for myself. I hope my letter offers a model of how to exit for other people as well. And frankly, I need the closure.

This isn't easy. You, Career, have been by my side my whole adult life and have served as a large part of my identity – although you, Career have never defined all of me. While we were together, we had good times. I thank you for all of it. Let's look back and enjoy some moments before moving forward.

What I loved

Learning. I love to learn. No one is ever done learning in technology. To be good in the field, you have to love to learn and keep learning. In a field that was so new when I joined in the mid-1980s that there were few books, and in a niche career path such as software testing that didn't even have newsletters, how do you learn? The internet didn't even exist in common practice. I had to find information in many ways and often on my own. I would have read the online help file, but when I entered the field in 1985 I was the person writing the online help or product documentation! Back then it was called the "product manual." Yeah, I wrote the manuals, and I wrote the online help for several software products.

Intellectual inspiration. I was in love with technology. If you know me, can't you see that gleam in my eye—the thrill of the software bug pursuit? It's quite addictive. My bedside nightstand was covered with tech books and magazines, and my podcast catalog reflected the same. The technology field was and remains a delightful, relentless beast of an engine that never stops, never quits, churns every day to find the new. The thrill of being part of the field was intoxicating.

Emotional satisfaction. Learning fed my brain. I had a deep gratitude that my brain was being fed. It mattered to me that I could apply my smarts to my work and that my knowledge made a difference both in the outcome and how I felt about my work. I was so damn happy to have a thinking job.

Variety. I worked on software for different industries, from banking and manufacturing to medical, and more. I worked on e-commerce when it was hot and new, and then I worked on mobile when that was hot and new. I've seen a lot of hot and new grow old, and then I chased whatever was next. I made my peace with this many years ago when I traded stability for variety.

Travel. Software testing gave me the opportunity to work across state lines and to speak at conferences internationally, providing opportunities to get to know people all over the country and the globe. I've had years when the number of currencies I was paid in made my taxes complicated but my life and work fascinating.

Flexibility. Whether in a corporate setting or, later, through having my own consulting business, you often provided the opportunity to work remotely, Career. This flexibility eased commuting and enabled me to be around while I raised my daughter post-divorce and also later when my parents were aging and ill. When I traveled to speak at a conference, remote work arrangements meant that I was still able to earn my project income by working in hotel rooms and on airplanes. Yes, it was a lot of juggling, but the flexibility made my whole life, not just my work life, function well.

Personal Development

I have many people to thank for their support and, with their help, I've built an impressive set of professional skills. It's funny how one acquired skill leads to the next in a virtuous cycle of what feels like good fortune but what, as a good friend always reminded me, is luck fueled by plenty of long hours and hard work. Career and colleagues, you sharpened my abilities in:

Asking questions. My degree is in Journalism, and people often ask how that background helped me with computer software. I think, "You're kidding, right?" Journalists learn to research, ask questions and do enough legwork to completely wrap their head around a topic. I took only one computer class in school, so clearly research and learning skills were key. But it was you, Career, who taught the master class. Through you, I learned how to ask multiple people the same questions to compare and contrast the responses, looking for discrepancies and gaps in information. I knew that where information would break down was where I would likely find breaks in software, too. I learned to interview people in full Q&A sessions as well as on the fly when I could squeeze in one or two questions at a time.

I became adept at asking hard questions in soft ways—to jingle information out of people without making them feel interrogated or think, “There’s that kid again, bothering me for more information.” I liked asking questions so damn much that I recorded a webinar called *The Art of Asking Questions* and, honestly, it was one of my best bits of work. Yep, I’m proud of that one. I was a perennial student.

Listening. Even when I wasn’t asking, I was listening. I got good at listening at lunches, in hallway conversations and during those moments before meetings begin when team members whisper questions to each other. And I especially perked up when software developers asked questions, because it didn’t take long for me to learn that when developers are confused, their code is going to be confused, and I’m going to catch my best bugs in those spots.

Playing to learn. As you know, Career, questions will get you only so far in the testing game. I also “played” with software—not in a silly, goofy way, but playing to learn, to experiment. I tried this or that and watched what happened. Research with no charted path was great fun.

Detective skills. Wandering and playing to learn left me feeling like “Nancy Drew, ace detective,” a point I tried to make in Ohio when I gave a keynote on the topic. Nancy would learn a fact, deduce another fact and gather all the information to devise a cohesive solution to the mystery at hand. I did the same thing with software; I just didn’t need a flashlight to do so. This all had practical application. Years into you, Career, when my daughter had an illness or my parents had health issues, those same investigation skills came into use. Essential skills of observation and pattern recognition are mighty helpful in figuring out illness. And asking questions in a soft way is a perfect approach to use for quizzing doctors. It was great to see what I learned through you, Career, help me in a practical way in my personal life.

Speaking. I could never say farewell to you, Career, without extending my appreciation for my time as a speaker in the field. Holy cow, I didn’t see that one coming! That I could become a geek sitting and learning and being shy—that wasn’t so surprising. But that I would grow and evolve to be able to stand in front of 1,000 people and comfortably give a decent keynote—that has been a huge surprise. Speaking gave me confidence. Speaking grew me professionally. Presentations made me think my thoughts long and hard enough to get to the clarity needed to then share my thoughts with unknown people in large settings. Being able to speak to many people also helped me to handle the smaller, more typical settings of, say, a meeting of 20 people.

Writing. Through you, Career, I had or created opportunities to write and blog. I had the good fortune to be published throughout my career—about a dozen articles in several magazines or newsletters and, once, a chapter in a book. It’s called *Beautiful Testing* and is sold on Amazon. Given that I went to school for journalism and English, these opportunities nourished my soul and gave me enough satisfaction with writing that I felt it was okay that I gave up a writing career for a computer career.

Running a business. When I became a solo consultant, I claimed the rights to mastering my future in choosing which contracts, clients and a whole array of other variables I would permit into my professional life. I was so busy doing the work and learning at the same time how to incorporate, purchase corporate insurance, invoice and handle billing that I didn't even notice the business skills I was acquiring.

Standing my ground. Over the years as my shyness wore off, I learned to argue. To defend myself and my point of view. To engage in debate. To enjoy layers of discussion not just to win or show off. When I pivoted into solo consulting, that personal sturdiness helped with contract negotiations and the occasional challenging client. (Most of my clients were great.)

Reasons to Go

So why am I leaving?

Long hours. I would barely lift my head from the computer screen and place my dinner order night after night as I burned to keep going, keep running—well, mentally running, as I would physically stay seated for long spells of time. I spent much of my time so deeply absorbed mentally with you I didn't even know I wasn't moving physically at all—just hunched in a chair, staring into a screen. I can't sit like that for you anymore, Career; I see now that those long days, day after day of sitting, were not so healthy. In my younger years, my body would quickly recover from our long-seated sessions, but that is not the case anymore. It hurts to sit for hours and, after 30+ years in computer software, my eyes now almost always squint. I need glasses for just about everything, even to read my damn phone—I particularly hate that. I'm much better now about periodically looking away from a laptop or mobile screen as my eyes just about demand that I do so. I obey my body's needs now over what my hyper-focusing mind sometimes still craves.

Getting older. Career, you are not forgiving as the years mount. Software is a high-energy field that demands the vigor and creativity of young professionals, who want to be surrounded with people who are like them in thought, experience—and age. They don't want to hang out with a parent figures. I have no interest in being bitter or getting angry at younger people entering the field as I leave. Staying in tech is like aging in general—it's more graceful and tasteful to let yourself evolve than to fight it and to try to keep being someone you once were but no longer are. I used to be one of the young ones, and I was young for a long time and then I wasn't.

Getting laid off. Sticking with you, Career, means that I've been laid off more than once for different reasons that could also leave me bitter, but that isn't how I feel in part because I took a sabbatical and have had think time not just reaction time. Instead, I've taken apart those experiences to see the message inside of those hard stops that says: it's time to go.

Change of identity. I don't see myself as a techie anymore. After these few years of soul searching, I am finally at peace with not identifying as a geek or a software person. That doesn't mean my time was wasted; it just means it's time for something else. Tech was a large part of me but not the whole of me, and now my tech interest has nearly entirely vaporized. I find that realization shocking. But I'm being honest here, Career. Gradually, dear friend, my interests shifted. Now my nightstand is covered in novels, memoirs and books on writing! My iPad is reserved for movies, and I haven't listened to a podcast on any tech topic in a long while.

So now what?

'Bye, Career in computer software, and hello, writing, my dear other friend! Now it is time for us to spend solid hours together. See, it all comes back; it all goes around. Months of sabbatical, fretful sleep and no paychecks, and I think I've got it now—with many thanks to my kind, patient and supportive husband while I've taken the time to sort through all this. Now I see that I need to write. And, I am not leaving writing again, not for anything. Not even if I don't get paid a dime and have to get a day job for the paycheck while the nights and weekends give me writing time.

As a child of 10 or 12, I wrote. It was not journaling; it was real writing. I wanted to be a writer. I didn't have the courage to write for a living in my 20s, 30s or even 40s, as I figured writers surely starve. And besides, I'd found software. And although I did want to write the Great American Novel, after years of living I now have plenty of stories without making up fiction—stories from what I've seen, what I've lived. I can write real-life stories that make people laugh or cry or laugh and cry at the same time. I'm writing a memoir and committed to completing it.

See, Career, I guess I do have a place to put my feet after all. You know what? It's going to be ok. Part of the challenge in ending my career in software has been feeling displaced at where to go, what to do next. I am too young and too filled with ideas to be idle. But now after a rest, I'm leaving sturdy enough to even go build another business, and that's what I'm going to do. I already know how to incorporate and how to handle a business—so much so that it almost won't matter what the business is. And that's pretty damn empowering, I'd say.

But you can feel sure, Career, that writing will be part of any business I run. And, although I remain an introvert, it seems unlikely that I won't present again—probably not at another software testing conference but in front of new clients, different organizations and various vendors or partners. As it turns out, you may leave a field, but you don't leave what you've become. And you don't become anything without opportunities and people. So, thank you, Career and colleagues, again.

I have an idea and a business partner. We've started but we are not ready to talk about it quite yet. This farewell to you, Career, must come before launching anything new. The separation is important. It is not always where you are going but where you have been; both parts are important. So, let's get this goodbye wrapped up.

Professional Community

I was part of and believed in sustaining a solid professional community. And if, during nearly every conference talk I gave in the testing community, I seemed like Sandra Bullock from “Miss Congeniality,” raging on and on about world peace, professional community and its importance let me highlight my gratitude for colleagues around the globe who felt the same and have been welcoming around even that part of me. Career, our field (well, my former field) is filled with lots of very good people who ran the race of technology with me for a long time. Thank you, dear friends. I’ll still be over here in Chicago wishing you the best.

What about my own peace? Achieving personal peace has taken me on a long journey, but now I am at peace and ready to go. I am also ready for new challenges. Writing a memoir. Building a business. And to spend more time and commitment to personal health and exercise.

Many thanks. Good to go.


Karen

Karen Nicole Johnson is a person who has the courage to reinvent herself and this year (2020) is going to be that year.

Her longtime professional background is in computer software with a specialty in software testing. She is also a person who has been a conference speaker for a longtime; both domestically as well as internationally. Karen loves to include stories whenever she speaks. She is a person who believes in the power of storytelling. Going forward Karen expects to focus more on writing.

More to come!



A photograph of a classroom scene. In the foreground, the backs of several students' heads and shoulders are visible as they sit at their desks. They are all raising their right hands, with some pointing their index fingers upwards. The students are wearing colorful shirts: light blue, red, orange, and green. In the background, a large, dark chalkboard is visible, with some faint, illegible writing on it. The entire scene is framed by a thick black border.

In the school of Testing

for your better learning & sharing experience

How to Read a Difficult Book

- Klára Jánová and James Bach

Have you ever had trouble reading a book about testing, although you consider yourself a big reader, a passionate tester and a curious person? I have.

I actually failed reading this particular book twice in the past two years. Well, maybe I didn't fail reading it, but I definitely did fail understanding it and learning the lessons that it offers.

I don't even remember when I first heard about Jerry Weinberg's *An Introduction to General Systems Thinking (ITGST)*. It was a long time ago. I am always on the hunt for new books to learn testing and people I respected frequently recommended this book to me. They said they have learned something new from it each time they read it.

I was so excited to get started, the first time, that I got the book in pdf and began reading it immediately after the purchase, on the bus, when traveling to work. But after completing the first few pages of chapter one and flipping through the rest of the book, it was clear to me that this was going to be a challenge.

Why was it hard to read? It's not just that there are a few English words (my native language is Czech) on each page that I don't understand. It's the way that the book is written. The sentences are long, and Jerry uses an academic tone. The subject matter is indeed systems thinking-- applicable to general situations-- but the examples that Jerry uses to illustrate it (at least in the first chapter) are mostly related to physics and computation, which means that in order to have a good understanding of it, I also have to be familiar with those topics.

I ignored those obstacles to start with and pushed through the first two chapters. But then I felt like I learned nothing from the book! Worse, it was like my brain turned off each time after reading a few sentences.

Since this was the first time that I encountered such a book and given that at that time I was 23 and just celebrated my second year of being a professional software tester after leaving the university, I came to the conclusion that maybe I'm not experienced and educated enough to read it just yet. I haven't given up on learning about systems thinking, but instead, I switched my focus to other books such as *Thinking in Systems* by D. Meadows and *Systems and Models* by H. Bossel.

I put Jerry's book aside and decided to come back to it later.

After a year, I gave the book another try. This time, I went through the first chapter and underlined each word that I did not understand; then I translated those words, typing it in my native language right into the book. Instead of reading a whole chapter at a time, I only read about ten pages. However, it did not help very much – in retrospect, I think this was partly due to my ignorance of learning more about the physics topics, before trying to make sense of what the book was trying to showcase with those examples. And ten pages were still too many at one sitting.

So the book with the treasures ended up being neglected in my library, again.

This year, I got interested in the applied epistemology topic from 'A Testers Syllabus' by James Bach. Since I did not know which resources (books in particular) to use to learn more about it, I contacted James personally. We had a chat about what epistemology is and how it relates to testing. He recommended three books to me – The Philosophers' Toolkit by Julian Baggini; Gödel, Escher, Bach by D. Hofstadter and the most important book to him, he said, was - surprise, surprise! - An Introduction to General Systems Thinking. But, in my mind, the chances of learning more about epistemology just dropped to the minimum. How was I going to learn anything about this topic when I already failed reading that book two times?

I shared my story about the problems I had encountered when reading the book with James, and he said that he himself had read the book in increments of a few pages at a time, followed by thinking through its implications. He said he got so many ideas when he read it that his mind would overheat. Maybe the problem I was having with it wasn't about me, but rather an effect of how much Jerry had tried to pack into the text. So, James suggested that I should also take it in small pieces and then discuss the content with him. I was hesitant for a second because I was afraid of failing again. But this was my best chance to finally succeed.

My Process for Reading ITGST (Third Attempt)

1. I read one to three pages in a sitting.
2. I underlined each word or phrase that I did not understand and translated those words or parts.
3. I highlighted phrases and sentences that I considered crucial.
4. I read each sentence multiple times, making explanatory notes in the margins about it (also in my native language, when needed).
5. I made an attempt to understand the topics that were used to illustrate the systems thinking problems such as the square law of computation, mechanics and Newtons' laws of nature.
6. I wrote a summary of each subchapter after I read it. When having trouble with understanding the bigger picture, I waited a day or two after reading this subchapter and then came back to it and read it again, and only after that wrote down the summary. I also thought about how the insights can be applied to testing.
7. After having written my summary, I sent it over to James. He read the same part that I had read and he commented on the inconsistencies between our understandings. This usually turned out into a discussion which follow.

5. I made an attempt to understand the topics that were used to illustrate the systems thinking problems such as the square law of computation, mechanics and Newtons' laws of nature.

6. I wrote a summary of each subchapter after I read it. When having trouble with understanding the bigger picture, I waited a day or two after reading this subchapter and then came back to it and read it again, and only after that wrote down the summary. I also thought about how the insights can be applied to testing.

7. After having written my summary, I sent it over to James. He read the same part that I had read and he commented on the inconsistencies between our understandings. This usually turned out into a discussion which follow.

Discussion Examples

Example 1 - 'Complexity of the World'(pages 1-2)

Klára's summary:

Precise mathematical results don't necessarily have to matter unless it ties to some proof.

These measurements are more often used to manipulate than to use them to prove something.

Mathematics can be used to describe a phenomenon in a way that might seem spectacular, but unless there's logical proof behind it, it doesn't have to be very helpful.

The problems that science is able to solve have consequences that humans are not able to predict. That relates to the progress made in the sciences, where knowledge came in very slowly during the ancient times (and before ancient times), in the range of thousands of years, but has sped up from that point on. We know that by the human actions, there will be some second-order effects (consequences) to the existing state, but we're not able to predict if it's going to be a change for the better or for worse and the precise magnitude of the change.

The purpose of general systems thinking is to help humans reveal the complexity of the world.

James: What do you mean about proof? In other words, what do you think that is about?

Klára: I thought they meant formal proof. Mathematical. But I think it could also be a different kind of proof than just formal proof.

James: Let's say you have two apples and you add two more apples, how many apples do you now have?

Klára: Four.

James: How did you figure that out?

Klára: By applying the sum operation. $2x+2x=4x$ where x is an apple.

James: Is that a precise mathematical operation? It looks sort of like one, to me. is it exactly 4? is it 4.000000000? Seems like that is what your operation gives.

Klára: You did not mention floating or double precision so I went with integers. The answer is exactly four.

James: Okay, then. So.... you said that doesn't mean anything without a proof. What's the proof? Did you think about a proof? Did you start with a proof?

Klára: No

James: Are you saying you don't agree with the idea that proof comes first?

Klára: No, it doesn't come first

James: Do you think you understand what the writer meant when he wrote that?

Klára: No (smiles)

James: I suspected that you didn't. because you copied his word "proof" instead of putting that into more useful terms. I will rephrase what he's talking about. "It is pointless to calculate things when you don't even know what you are talking about."

In terms of the apple example, I asked you to add two apples to two apples, but you made a bunch of assumptions about what I was talking about. You didn't check these assumptions

you are apparently not worried about misunderstanding my question. (Which is okay, of course, unless it's the sort of question that really matters.)

But the assumptions might matter. For instance, if you were to add two anti-matter apples to two ordinary apples, the result would be zero apples (because they annihilate each other). Or if you add two apples to two other apples that had been sitting there for 3 million years, you would have only two apples (because the old apples would have disappeared).

Klára: Yes, I was actually thinking about what kind of apple that could be (the fruit, or some reference to Apple's devices), but I didn't ask because the result I was thinking about would not change anyway. I didn't think about the rest of it, unfortunately...

James: Whether you thought about all the different possibilities doesn't really matter, in this case. What matters is what you actually were thinking. All I'm saying is that your equation is predicated on certain premises that were in your mind. Your answer is true within those limits. Thus, in order to communicate usefully and think effectively, you need to know the context of the numbers and equations that you use.

In summary, this is how Jerry is starting out his book: we need to understand the structure of a system in order to reason effectively about that system. Structure dominates all other things. A small change in structure may create a huge difference in behavior or nature or appearance or evolution.

Klára: So what is structure?

James: My most general answer (based on reading ITGST) is that structure is any pattern that persists in time and space.

Klára: Is it just one pattern, or does it include different 'sub-patterns' too?

James: Any pattern. A set of patterns is a pattern. A pattern may consist of a set of patterns.

But there is a more specific meaning in this case. A system consists of a set of things in a meaningful relationship with each other. (I'm using my personal definitions but I believe they are consistent with what Jerry wrote.)

So the structure of a system would mean persistent patterns in the meaningful relationship of elements of that system to other elements. That may include relationships of a multiplicative or exponential nature. Thus, a small change in one part of it could disable or magnify any given behavior. This is just like software! You are used to this.

You could read the first chapter as implying that you can't assess the quality of a software product just by carefully counting the pixels on a window. You have to understand the meaning and the relationship amongst the pixels, what drives the pixels, you need to know the functionality, perhaps the code itself.

Klára: That did not occur to me at all, but it makes sense now.

James: He explains in those pages that knowing how to identify and navigate patterns and relationships in and among systems allows you to learn more quickly. Try re-reading the first pages with that thought in mind, I bet it will make more sense to you.

Klára: *"Measurements and equations are supposed to sharpen thinking, but, in my observation, they more often tend to make the thinking non-causal and fuzzy."* Could "measurements and equations" be changed to "automation tools" in the context of software testing?

James: Yes. or even more broadly, "best practices." Any adherence to practice that becomes separated from reason and instead driven by habit.

Example 2 - 'Mechanism and mechanics' (pages 3-5)

Klára's summary:

The success of physics as a science is so great because it reduces the set of all the possible conditions that could be applicable to some object to only some conditions, which are sufficiently defined and under which the behavior of the object evinces regularities.

It does not make sense to try to compute all the equations (even by approximate methods) of mechanical systems that have too many parts, because we don't have enough time to solve it.

For that reason, we have to reduce the big mechanical system to a more simple one.

The importance of the parts of a system depends on the context of the question that the scientist wants to answer. Even if a certain part of a system is very small compared to the others, and it might seem insignificant to some, it might actually be the most important one for a certain scientist that deals with a context proper for researching this smallest part.

How this relates to software testing:

1. Observing the regularities in the system at first might help to learn about the system; what it does, why and as a starting oracle for future tests.

2. When engaging with a new product, it might be helpful to map out the features and behavior, but not a lot of time is needed to spend on the subtle details that relate to an area that is probably not that important.

3. Start informally, formalize the testing later (if needed).

4. The importance of various areas of the system is dependent on the context - to what questions are we seeking answers to?

James: Just as a tree has a trunk, which separates into limbs and ever-smaller branches and twigs, we can look for the “trunk” of a system we seek to test— the core part. So we need to be able to identify the stuff that matters more and the stuff that matters less. Identify the trunk first, then you can deal with the lesser details.

Klára: Exactly, but apart from that, the importance is also affected by information that we gather from the stakeholders or developers, right?

James: Sure. There are different factors that go into what matters most.

Klára: I was wondering if this part:

“The success of physics as a science is so great because it reduces the set of all the possible conditions that could be applicable to some object to only some conditions, which are sufficiently defined and under which the behavior of the object evinces regularities.”

is also applicable to test techniques. I think it is. I mean, that various test techniques are useful only under some specified conditions.

James: Definitely! That’s the same concept I was just talking about. Equivalence classes is all about that.

Klára: *"Many philosophers thought, with Laplace, that given precise observations on the position and velocity of every particle of matter, one could calculate the entire future of the universe. Although they realized that they would need a large computing machine, they lacked even the smallest computers"*

What do you think is meant by 'calculating the entire future of the universe'? The only thing that I think it could mean, is that the scientists could calculate the bodies exact locations at a particular time?

James: Not only exact locations but the result of every chemical reaction. In other words, the exact state of the future universe at any moment in time, as if running a video in fast forward.

This is a thought experiment, of course... it's not physically possible, but depending on one's model of the universe, it is or is not possible in principle.

Klára: *"Newton needed all the simplifying assumptions, explicit or implicit, he could get away with, just as physiologists and psychologists do today."*

How do we draw a line to what's an important thing to consider and what not?

James: The things that are important to consider are the ones that make a difference

Klára: But we don't know about all the things that could make a difference upfront right?

James: Okay, so what is your real question? Are you asking how to know about all the things that could make a difference?

Klára: No, I don't think that's possible.

James: Then what is your question? I mean, are you just trying to confirm that we can't know?

Klára: Yes.

James: Technically, we can't know for sure, but usually that doesn't matter very much. You make calculations all the time without knowing things for sure. For instance, you calculate how long it will take you to get to work. When you do that, there are millions of tiny tiny tiny unknowns but you don't worry about them.

A better question is not "how do you know," but rather "how do you decide" what matters?

And the answer to that is theory. You have a theory about the world and you modify that over time. This theory tells you what variables matter. Theorists are people who build concepts that help practical people get work done, but we are all theorists to some degree.

Klára: So in testing, what is this theory? Is it the mental model?

James: Well tell me, if you test emoticons in Skype, do you think it's important to vary the kind of tea you are drinking at the time? I bet you don't. But why don't you? Your theory of the world says that what you happen to be drinking cannot affect the behavior of emoticons in Skype. This is not simply logic, it's using logic in conjunction with theory.

Klára: That sounds more like experience than theory

James: Experience is not theory, at all. Experience is what happens to you. People often use the word experience to speak of the learning you have gained from what happens but that learning is not experience itself.

Klára: Exactly, experience is different than theory. what I'm saying is that the situation described with the tea seems more like based on experience than on theory - I observed that the variation in non-alcoholic drinks does not affect my testing.

James: That's theory! When you say experience you are just telling me where your data came from. That data has no theory "in it." Nothing about drinking tea tells you directly that it doesn't affect emoticons.

Klára: So what is 'theory' then?

James: Theory is a set of concepts that tells you what matters and how it matters.

Klára: Oook.. That makes sense now.

James: You have a theory that tea doesn't matter. And that, in turn, is based on your theory of matter, energy, computers, etc.

Klára: Yes, I agree now.

James: Does the exact position of the sun affect your testing? I once experienced that sunlight affected a test I was running. When I realized it was happening, that wasn't just an experience that showed me it was possible. It was an experience that made sense in theory, because the mouse had an optical sensor in it and there was a crack where sunlight could get through and the sun was shining through my office window. Light shining on a sensor designed to detect light is bound to affect that sensor, which is bound to affect the computer connected to that sensor.

Klára: That's a good example.

James: Here is the big difference between experience and theory; *you don't have to experience the thing I just spoke of in order to accept that it is plausible*. Because you already know about sunlight and sensors and computers that read sensors. I told you the bare details of it— the details that mattered. I didn't tell you about the kind of grass outside my office window and you don't feel the need to ask about that.

And now you know why propaganda and "fake news" and conspiracy theories can be so powerful, because when a story sounds *plausible* (i.e. when it fits your theory) and when a person *wants* that story to be true or is *fearing* that it might be true, then that person may be eager to believe it *regardless of their experience*.

Conclusion

As you can see, the discussion is an integral part of my current process of reading the book. That's the point where I learn the most. Even if I do a really good job figuring out what Jerry meant with his words, I gain much more insight when discussing the content with another person - especially when the person is James, who had spent time with Jerry and is able to discuss on my follow up questions on the systems thinking topic, the answers to which Jerry did not explicitly state in the book (such as his definition of 'structure').

You will find that the benefits of this social method of reading will be different, based on who you read the book with;

- when reading the book with an expert, he can guide you through the material and provide valuable insights that will help your understanding of it
- when reading the book with a fellow colleague or a friend who is new to the topic of the book just like you, your discussions might not be as deep as when you're discussing it with an expert who's able to switch to lecture mode, but you will still get some mental exercise by putting your understanding of the material into your own words and it can help each other's energy.

Additionally, the novice readers might find it hard to discuss the books content on the spot without having written down a summary or at least some notes, that they can refer to during the discussion. If the language of the book is not their native language, this can cause problems as well.

Going through this process of reading the book with James, I found out that

- there are different factors that influence the level of knowledge and the benefits that one gets from reading a book and speed is just one of them - there actually needs to be a balance between the speed of reading and the depth of understanding and knowledge; (in other words, there are books that take time to read and the fast speed of reading is not always the most important factor)
- while it's almost always the case that discussing the content of a book with other people helps, for harder to read books like ITGST it is even much more important to do that



James Bach is the creator of Rapid Software Testing methodology, co-author of Lessons Learned in Software Testing and author of Secrets of a Buccaneer-Scholar.

He is an avid student of Jerry Weinberg's work, and was also co-host of the first Amplifying Your Effectiveness conference, and lead editor of the Amplifying Your Effectiveness book, working with Jerry.

Based in the Czech republic, **Klára** considers herself a context driven tester and practitioner of the Rapid Software Testing methodology working at TestLauncher, the company that provides advanced software testing.

She's an instructor of the Black Box Software Testing classes at the Association for Software Testing, a peer advisor for the Rapid Software Testing Applied class and a member of the editorial board at Tea-time with Testers, who loves animals, piano playing and Charles Bukowski.



Context-driven Approach for Testing Complex IoT Systems and Devices

- Milena Lazarevic and Dejan Nikolić

The IoT story

Since the internet of things (IoT) is very popular nowadays, this is a great opportunity to share some interesting experiences of testing the complex IoT system and devices.

The idea of having all the devices around us wirelessly connected has been alive for almost a century. It was one of Nikola Tesla's visions of the future. The problem which kept this idea from coming to life is the cost of hardware needed to upgrade the devices to be connected to the Internet. With IPv6 in existence, every atom on the Earth can be assigned an IP address and that can be done with a 100 more planets, as Steve Leibson defined. So, the new rule of the future will be "anything that can be connected will be connected".

Some of the examples of the IoT systems are smart homes, wearables, smart cities (from traffic management to water distribution, to waste management, urban security, and environmental monitoring), industrial internet (manufacturing, logistics, oil and gas, transportation, energy/utilities, mining and metals, aviation), connected cars, connected health, smart retail, smart supply chains and smart farming. Devices that are used in IoT systems are widely ranging from simple sensors to very complex devices such as cars, vending machines, smartwatches, traffic lights, humans, animals...

What is context-driven testing?



Human-centered
approach



Practice is continuously
re-evaluated



Continuously applying
and learning new skills



Product is a solution
to an existing problem

IoT systems are complex environments formed by different types of sensors and devices, different types of communication protocols and applications who used different types of data. The main challenge is how to test this complex architecture, what test procedures and good test practices can be used in testing complex IoT solutions. Context-driven testing school of thought represents the answer, as it promotes good practices in context.

The context-driven school of testing was declared on the 21st of November 1999 by James Bach, Cem Kaner, Brian Marick and Bret Pettichord (with the support of other colleagues).

Context-driven testing is a human-centered approach. People with different skill-sets and knowledge who are working in a complementary team are one of the greatest value for the project.

Working on a project, judgment and skill are always accumulated and re-evaluated against the project context.

There are good practices in a specific context, but there are no best practices. Good practices are always re-evaluated since projects unfold over time in a way that is often not predictable. In one situation a good practice can give great results but over time, as the context of the project changes, that practice may not bring the anticipated results anymore. The context-driven test approach puts focus on awareness to do the right things at the right time and to test product effectively.

The product is a solution to a problem. If the product doesn't solve the problem, it simply does not work. In the context-driven test approach, the focus is on the context of the product as a solution.

Why do we need context-driven testing?



The value of context-driven testing is the way testing is organized around the product context.

There are two products (picture above), one is a smartwatch for measuring steps, heart rate, activity, etc and second is a medical infusion pump for monitoring body fluids and infusing drugs in patients. Both products measure body parameters but the context in which those two products are used is different.

Medical infusion pumps are used in medical care and hospitals. Smartwatches are used in everyday life. Test practices that we use for testing the smartwatch could not be used as good practices for testing the medical infusion pump and vice versa.

For testing the smartwatch, the focus would be on functionality, usability, connectivity, etc. For testing medical infusion pump focus would be on testing security, standardization protocols, etc. In conclusion, context-driven testing puts products context first, around which all suitable test practices are organized.

How do we use context-driven testing?



Information in context

In the context-driven test approach, the main purpose of testing is to provide information to different stakeholders. The context of the information is not the same for all stakeholders. For example, the test report for the management is usually not the same as the test report for the clients.

"X" shaped tester

Testers should possess a different set of skills and knowledge. Other than good technical knowledge, testers should have good management and communication skills in order to be an advocate for different stakeholders and to put the information in the right context. Of course, one tester doesn't know everything so the focus should be on a complementary team that covers all necessary knowledge and skills.

Role-playing

Since the product can be used by different types of users, testers should simulate the behavior of those users by role-playing while testing the product. This way, product context would be evaluated in the right way.

Session based testing

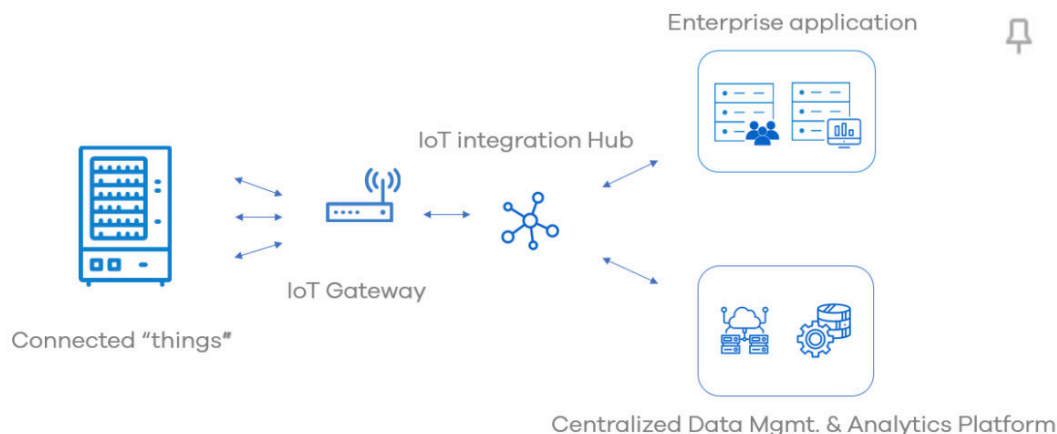
Session based testing is in the main focus of the context-driven test approach since a lot of the important problems can be found following test ideas and improvising during testing. Basically, session based testing is exploratory testing activity recommended to be organized in sessions with a specific time frame (see session-based test management by James and Jon Bach) in order for testers to have focus during testing.

The intelligent vending machine as a complex IoT device



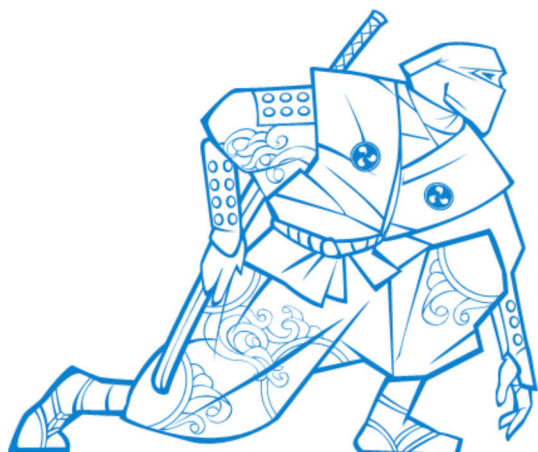
Let's look at an example of a smart vending machine as an IoT device.

An intelligent vending machine has a mini PC inside that drives the software, a big touch screen with the sales application, and part of the software which detects demographic profiles of the users (e.g. adult male, adult female, senior male, etc). All the hardware devices and sensors are connected to the mini PC and can communicate with the software. The machine is always online and can send all the data (e.g. current state of the inventory, state of hardware devices inside, transaction details, user interactions, etc) to the backend system. All of that data is used to create one business intelligence model and lead to a smart retail solution.



How do we cope with testing challenges?

Testing the software with the hardware is very sensitive and it needs different kinds of testing skills.



Testing as an R&D process

Testing has been included in the developing process of this product from the beginning. It is an important part of the R&D process. While searching for the perfect mini PC for the machine, we conducted some performance testing on numerous PCs to evaluate the behavior of the CPU consumption, the heating of the PC and the performance of the sales application while the PC is connected to all the peripheral devices inside the machine. The degradation testing of the hardware devices was a way for the software to learn how to cope with hardware issues. For example, the test when the coin is jammed inside the coin payment device taught the software how to continue working after this situation had happened on the field. Testing the coin device as a payment device can also be very interesting. Situations

such as device disconnected from the integration board, different coin sizes and denominations, coins jammed inside the insertion unit and dispensing unit, are all very valuable when creating a test plan for degradation testing of the payment process of the machine.

Hardware dependency

In order to make sure that the testing will be performed correctly, all of the hardware components (e.g. cables, sensors, devices, etc) need to be in the proper state prior to starting the testing. The order of action when investigating one issue found must start with the verification of all the hardware inside the machine. For example, if coins are not accepted in the machine, it can happen that the coin payment device is in an incorrect state and that the issue is not in the sales application software. So first, we need to remove all the possibilities that the hardware parts are corrupted and after that report an issue to the software development team.



Limited resources

Since the mini PC has limitations in terms of processing power and the software for facial recognition is working offline and needs a lot of processing power, we need to monitor the CPU consumption while testing the sales application.

Network interruptions

The internet connectivity performance can depend on different hardware implementations (e.g. Wi-Fi module, 3G, 4G, LAN, etc) and the location of the vending machine (e.g. the metro station below the ground, office in the building with walls as Faraday's cage, etc). Our tests need to cover network interruptions of the machine in every moment of the software lifetime (e.g. during the product purchase, while sending events from the machine, while receiving new media (images, videos) on the machine, etc).

Different types of users of the system

Role-playing is very important in the testing of our system. There are three different parts of our system; the sales application (used by end consumers of the products inside the machine), the maintenance application (used by technicians that maintain and refill the machines) and the backend portal (used by operators in the office and the business people). In order to make sure that the needs of those users are satisfied with this software solutions, we need to approach the testing with certain stories such as:

1. Sales application users can be kids who like to play with the touch screen (the software needs to support lots of user interactions with the touch screen), people who are in a hurry, waiting for the train at the train station (transactions need to be very fast), elderly people who are afraid of new technologies (transactions need to be very simple), etc.
2. Maintenance application users can be people who refill the machines and who do not need to know anything about the software, so the application must be very simple, such as entering just a few codes.
3. Backend portal users can be office people who monitor the state of machines on the streets (the portal needs to have real-time information about the state of hardware inside the machine and the state of the inventory; all the proper data need to be available on the portal all the time), business people (they need to be able to get all the sales information and different relative statistics from the machines, and all of this data needs to be accurate and properly filtered).

Security issues

Since the machines are standing 24/7 unattended in public, they need to be on a certain level of security. Regarding the hardware, the machines have locks and security glass. All of the communication with the backend system and the cloud is encrypted. But sometimes, people think of different ways to get their products for free. There was one situation shared by a customer in production. When the people who refill the machines opened the door, they found a cat inside. Someone inserted the cat in the output tray of the machine and the cat jumped onto the plate with products. While the sound that its claws made while she was playing around with the snacks inside the machine amused her, several products fell down in the output tray and the customer got his products for free.



All of those different aspects of hardware and software functioning together, demand from the tester's mind to always be broadened and search for new approaches, while constantly asking the question "What if...?"

Future challenges of testing IoT solutions



In the future, testing the complex IoT solutions will bring many opportunities regarding developing new test procedures and incorporating IoT technology in testing processes and testing tools. But also, the future brings many challenges regarding standardization in IoT testing, test automation, testing scope, and connectivity challenges.

Standardization

Standardization in testing IoT solutions is still in its infancy. There are no test methodologies to follow, only good practices. In a way, it is a good thing because it gives flexibility in testing and the opportunity to try different test practices. On the other side, standardization could give a starting point and a guide on how to implement test procedures and good practices in different projects.

Test scope

Defining the testing scope for IoT systems with maximum possible coverage will be a challenge in the future. Possible solution is to focus on the end-users and how they use IoT systems. That information could be used to create test coverage that covers the necessary functionalities of IoT systems.

Connectivity

Without connectivity, there are no IoT systems and the main challenge in the future will be testing different types of connectivity, different signal strength, and different communication protocols. Solution to that challenge is to continue using network simulators for simulating different network conditions and network signals.

Automation

Since IoT systems include hardware and software, automation will be the greatest challenge for this type of system. Today, we have automation frameworks that can help testers, but those can automate only the software part of IoT systems. Automating hardware and human interaction will demand to create automation platforms which include hardware simulators running together with automated scripts.



Milena Lazarevic is head of testing in a young software development company called Invenda based in Novi Sad. After completing a Master's degree in Mathematics at the University of Belgrade in 2012, she started testing various vending and transportation software products for international clients through a Serbian consulting company. She has main experience in the ticket vending industry, where she has been using multiple approaches for testing complex mixtures of hardware and software.

Searching for new challenges and excitement, she joined some of her former vending colleagues at Invenda where testing is an integral part of the R&D process in building a product that brings life to automatic devices.

Dejan Nikolic is Test developer at Invenda Solutions.

For him, quality is never a coincidence but the end goal and primary expectation of software testing. Consulting for 6 years, Dejan developed and implemented testing strategies for government and institutional clients in the transportation sector, with a focus on ticket vending and mobile applications. Through his career, Dejan kept both feet in manual testing and automatization, developing testing approaches used across multiple software and hardware solutions. Dejan's main passion is context-driven testing and funky and challenging intellectual process which comes with good software testing.

Dejan completed a Master's degree in Computer and Economic science at the University of Novi Sad in 2015.





Sharing is caring! Don't be selfish ;)

Share this issue with your friends and colleagues!



Happiness is....

Making home-office better by reading about testing!!!



Like our FACEBOOK page for more of such happiness

<https://www.facebook.com/TtimewidTesters>

T ' Talks



T. Ashok exclusively on software testing

Dissecting The Human/Machine Test Conundrum

Summary

It is common to see testing discussions veer into a dichotomy of “manual vs automated testing” and how the latter is indeed the order of day today. Sadly I find the discussed seriously flawed. In this article I dissect these the way we test as being human-powered and machine-assisted and outline an interesting way as to how the role of power of human and machine assistance is paramount to do testing smartly, rapidly and super efficiently.

Introduction

The way we test has been trivialized into two buckets of manual and automated test by general milieu. Firstly the phrase “manual testing” seems to connote a menial job, intensely labour oriented which it is not and therefore the phrase highly incorrect. Secondly the notion of automated testing seems to connote writing scripts and running it frequently to detect issues. What is forgotten is that once a test script uncovers an issue which when fixed makes this script as a health ascertained rather than a ‘bug finder’ and that test cases/strategy needs to be constantly updated to newer more ubiquitous issues.

Human-Machine testing NOT Manual-Automated testing

The more appropriate word would be HUMAN testing, as it connotes a combination of BODY-ily activity with INTELLECTual thinking powered by MIND. MACHINE as term is probably more appropriate in signifying an aid to test in a holistic manner rather than AUTOMATED testing, which seems to connote only build and execute.

HUMAN Testing = intellect+body+mind

Philosophically human is seen a composition of Body, Mind and Intellect. Using the same idea context of testing, I see the act of physically observing, hearing, doing, feeling BODY-ily activity while INTELLECTual activity powers some of the key activities of testing while MIND enables appropriate thinking.



The backdrop to dissection

To dissect the various the activities and understand what needs to be done HUMAN and what can be by a MACHINE, I am going to use the Test Activity list that outlines key activities in the lifecycle of testing:

- Understanding the System under test
- Strategising and planning the test
- Designing test scenarios, cases, data sets
- Executing tests including automation
- Reporting – issues, progress
- Analyse – issues, test progress, learnings

Now we will analyse the various HUMAN-powered and MACHINE-assisted activities for each key activity.

HUMAN-MACHINE Test Map

The complete HUMAN-MACHINE conundrum dissected on following image:

TEST RELATED ACTIVITY	HUMAN powered			MACHINE assisted
	INTELLECT	BODY	MIND	
Understanding	read, explore, relate, visualise, question, empathise, find holes, conflicts, ambiguities, draw, jot notes, code analysis	observe (see) listen (hear) do feel (touch)	-mindsets- logical creative -traits- curious suspicious value oriented -habits- disciplined persistent striving to perfect revise constantly move rapidly -being mindful-	visual drawing tools - charts, mind mapping, code analysis tools
Strategize & Plan	identify persona, EUTS, attributes/criteria, environments, hypothesise issues, setup baseline, estimate time, effort, budget, dialogue with stakeholders, skills/people			project, test management tools estimation tools, complexity analysers
Design	understand architecture, data flows, behaviour, generate scenarios, cases, data sets, probe structural aspects, enable testability			BDD tools, MBT tools, Test data creators, Combination tools (OA, Pair wise)
Evaluate	learn, unlearn, revise be repetitive, stay sharp			test scripting , frameworks, data creators, comparators, tools for specialised tests, log analysers, code probes, environment setup, schedulers, coverage , leak detectors ...
Report	being crisp and detailed, being factual and subjective being firm and suggestive			change, defect, test, project management
Analyse	visual representation lead to actions enable learning, decision making (technical, managerial & business)			charting, dash-boarding

Note that this is not be intended to be comprehensive filled with all tool aids, nor all the human activities as this map would lose its utility then! Use this as as aid to understand the HUMAN-MACHINE test conundrum and for heaven's sake STOP USING the phrases "MANUAL and AUTOMATED testing".

"It is time we recognized that it takes smart HUMANS assisted by MACHINES (really tools/tech) to test less, test rapidly and accomplish more"



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



Self-Criticism, the key for Continuous Improvement

Continuous Improvement is something we all hear and even talk about every day. I am just not so sure we understand how to really implement a culture of Continuous Improvement that can stick and provide on-going results for our teams and companies...

Continuous Improvement is critical in today's fast paced world. Our teams are expected to deliver features in a matter of weeks or even days, and so we are in the lookout for ways of making things a little bit faster, to cut waste even further, to achieve more with less people and in less time.

Obviously, at the beginning of the process, when things are really messy and the process is not even close to being fine-tuned, finding places to improve and achieving gains is relatively easy.

You usually start by streamlining things that bother everyone, these are also the things that no one minds changing to make them more efficient.

But once you "run out" of these easy wins, you need to start looking further into the process. This means implementing changes that improve the work of the many, but come at the expense of a few members of the team who need to invest more time and efforts from their sides.

This sounds trivial: We are all in the same ship and headed on the same direction, so everyone is willing to cooperate, right?

Well, not exactly...

Once people understand they are "the only ones paying the price for progress" you start hearing more arguments and rejections.

Some see the direction of the ship as something that can be argued about, and others may agree on the direction of the ship but disagree on the path to reach it. So change comes with more friction and less naturally to all.

Here is where a culture of Self Criticism can help your path for Continuous Improvement.

Self Criticism is the action (some may say the ability) of looking at oneself and find the things that we can change in order to improve. This action or ability should not be mistaken with lack of self confidence, that is a self destroying attitude, one that never results on any gains to you or the team.

Achieving an attitude of Self Criticism in your team is usually hard work. Don't want to discourage you, but it is better to be realistic about it.

You need to ensure first and foremost that the whole team personally and professionally trusts one another - if this is not the case some people will always feel others are trying to take advantage of them.

This step is not easy.

People are diverse, they each have different ways of working, of communicating, even of understanding situations. In short, each two people in your team will see almost every aspect of their day differently, and they will need to learn to trust one another.

Once this is in place and your team members can work with each other, you want to have a way to make the work of the team as internally transparent as possible. This way every person is able to see what the others are doing, and how "my work" interacts with theirs.

Again, this is also not trivial as many people are afraid of working openly and sharing the way they operate. Only once this becomes the norm of the team, will all team members agree to do it openly.

Finally, you want to have a method to bring forward ideas and activities that can be improved, and to do this without pointing fingers or playing blame-games within the team. This is where Self Criticism allows the team and every individual within it to bring forward changes from their side for the betterment of the whole team.

The main problem here is that we are not talking about a change in process, but more of a change in mindset and company culture.

These changes are harder to achieve and take more time. But on the other hand, once you have made these company culture changes, you start seeing gains from areas that were not planned or even expected. So it is an even better investment in the long run.

Joel Montvelisky is a Co-Founder and Chief Solution Architect at PractiTest.

He has been in testing and QA since 1997, working as a tester, QA Manager and Director, and a Consultant for companies in Israel, the US and the EU. Joel is also a blogger with the QA Intelligence Blog, and is constantly imparting webinars on a number of testing and Quality Related topics. Joel is also the founder and Chair of the OnlineTestConf (<https://www.onlinetestconf.com/>), and he is also the co-founder of the State of Testing survey and report (<https://qablog.practitest.com/state-of-testing/>). His latest project is the Testing 1on1 podcast with Rob Lambert, released earlier this year - <https://qablog.practitest.com/podcast/>

Joel is also a conference speaker, presenting in various conferences and forums world wide, among them the Star Conferences, STPCon, JaSST, TestLeadership Conf, CAST, QA&Test, and more.





What's making News?

Testers, Storytellers, and Cultural AI

- Davar Ardalan

QA and Testing leaders engage in emergent trends and practices in software testing and quality including the future of cultural intelligence in AI.

How can testers prepare for the future of AI, testing, and personalization? That's what we workshopped in November 2019 at [ConTEST NYC](#) led by Test Master Academy's Anna Royzman in New York City.

As the Founder and Storyteller in Chief of [IVOW AI](#), I was invited to showcase our research in cultural IQ as well as our global dataset challenge — [An Algorithm for Stories on Women](#). The focus of our workshop was how to prepare for the emergence of cultural intelligence in AI.

As an automation architect for Discovery (USA), Brian Saylor oversees the planning and strategies for testing and data integrity automation for websites like FoodNetwork.com and TravelChannel.com. His group came up with these two ideas:

1. Build standardized datasets which include data from multiple cultures that can be shared with companies. In this way we will have easy access to bias-free data for training machine learning algorithms.
2. Build a tool that can test datasets for bias. When custom datasets are created for training, they can be tested to determine the types of cultural or other biases that may exist in the data.

Yes! For the field of AI and personalization to be effective and culturally relevant, we must create comprehensive datasets to nurture cultural intelligence in machines and even in social robots like Sophia of Hanson Robotics, who has traveled to over 30 countries. Each time Sophia travels, she meets people of all different backgrounds who talk to her. Collecting culturally prominent datasets in an efficient and scalable manner today is paramount to future commercial success of any AI solutions and products and inclusive AI.

Other testing leaders present included:

- Angie Jones - Automation Architect, Senior Developer Advocate at AppliTools (USA)
- Shesh Patel - The New York Times Engineering Manager
- Tanya Kravtsov - Director of QA at Audible
- Jason Huggins - Founder of Tapster Robotics (USA)

Adam Mardula is a Test Engineer at Business Insider in New York City. He workshopped with a team that suggested creating a sandbox / testing environment where testers can interact with the current AI technologies that are out there — like robots.

“In this case,” Mardula says, “we can potentially just have one component of the Sophia project available in the sandbox: Emotion tone interpretation.”

Mardula said testers could have the opportunity to speak to the robot with frustration, urgency, excitement, fear, happiness, and so on, and the sandbox should theoretically be able to provide the appropriate feedback about what was spoken. “In this way we can have an environment in which tests can be run. Enter a speech snippet spoken with some emotion, and the correct feedback about the emotional state of the speaker should be returned. The same results should be returned from speakers of different backgrounds, cultures, and even languages,” Mardula said.

It was exciting to see the enthusiasm in the room as teams from all over the world imagined this future. At IVOW AI, we have joined Test Master Academy as part of our global dataset challenge to crowdsource and make available open data on stories of women in history, culture, science and technology. This collection is necessary to provide critical historical context for the preservation of culture with an emphasis on the role of women.

Join our [crowdfunding campaign](#) and become a Founding Public today! This is the first of ten global AI and Storytelling challenges between 2020–2030 that will introduce stories to AI and advance cultural intelligence in AI systems. Future datasets will focus on:

- stories of healthcare and heritage;
- stories of indigenous people and nature;
- stories of people on the Autism Spectrum;
- and stories of refugees.

“We are excited to join forces with IVOW AI to make the first steps in bringing inclusiveness and quality to the future of our technological society,” says Anna Royzman of Test Master Academy. “This collaborative project will offer multiple opportunities for the global software testing community to learn and be involved in a hands-on project making a positive impact on humanity.”

The goal of the first dataset challenge — An Algorithm for Stories on Women — is for participants to develop an algorithm that can generate a character profile when provided the name of a prominent female in history, science, technology, or culture including folklore and myth. This algorithm is intended to scrape information from various sources off the internet and generate a character profile which includes a caption that is fewer than 100 words, and responses to metadata tags.

“Someday in the future my great-granddaughter will ask ‘Google, why do Indians wear a red dot on their foreheads?’ I want the answer to be truly reflective of her ancestry and include the emotions that I would feel in answering that question, rather than the one-size-fits-all answer: ‘It’s common practice to do so’.” - Aprajita Mathur, leading bioinformatics software test engineer and Senior Advisor to our dataset challenge.

If you’re interested in learning more, please contact us and consider contributing as one of our first [Founding Members](#). Your sponsorship and contribution will play a vital part in creating this pioneering dataset challenge.



Davar Ardalan is the Founder and Storyteller in Chief of IVOW AI, building cultural intelligence for AI.

A banner for the dataset challenge. It features a row of five historical women's portraits. Below the portraits, the text reads: "DATASET CHALLENGE", "An Algorithm for Stories on Women in History", and "#VoicesofWisdom". At the bottom is the IVOW logo, which consists of a stylized green and blue figure above the text "IVOW".

DATASET CHALLENGE

An Algorithm for Stories
on Women in History

#VoicesofWisdom

IVOW

Advertise with us

Connect with the audience that MATTER!



Adverts help mostly when they are noticed by **decision makers** in the industry.

Along with thousands of awesome testers, Tea-time with Testers is read and contributed by Senior Test Managers, Delivery Heads, Program Managers, Global Heads, CEOs, CTOs, Solution Architects and Test Consultants.

Want to know what people holding above positions have to say about us?

Well, hear directly from them.

And the **Good News** is...

Now we have some more awesome offerings at pretty affordable prices.

Contact us at sales@teatimewithtesters.com to know more.





Every Tester

who reads **Tea-time with Testers,**

**Recommends it to friends and
colleagues .**

What About You ?

our family

Founder & Editor:

Lalitkumar Bhamare (Germany)

Pratikkumar Patel (The UK)



Lalitkumar



Pratikkumar

Inspiration and Contribution:

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 – Debby Hudson

Editorial Board:

Dr.Meeta Prakash (India)

Dirk Meißner (Germany)

Klára Jánová (Czech Republic)



Dr. Meeta Prakash



Dirk Meißner



Klára Jánová

Online Collaboration:

Shweta Daiv (Germany)



Shweta

Community Partner :

Kiran Kumar (India)



Kiran Kumar

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

To get a **FREE** copy,
Subscribe to mailing list.

SUBSCRIBE

Join our community on

facebook.

Follow us on – @TtimewidTesters



www.teatimewithtesters.com

