WAKING TESTERS UP SINCE 2011

ISSUE #01/2024

TEA-TIME WITH TESTERS

AN INTERNATIONAL JOURNAL FOR NEXT GENERATION TESTERS

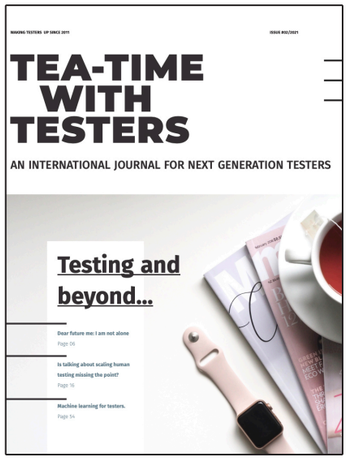
API Monitoring

Data Migration

Leadership

TEA-TIME WITH TESTERS

TEA-TIME WITH TESTERS



A NEXT GENERATION TESTING MAGAZINE

Thinking about testing and about testing our thinking

A lot has changed since we last interacted, hasn't it?

Well, it seems so to me. With each passing day, some new developments on the AI front appear in my news feed. The speed with which things appear to be improving (or new possibilities getting introduced) is likely to make many of us feel overwhelmed.

What does it mean for testers in particular? I would say it means a big window of opportunity and also a time for deep reflection.

The "Automate everything (blindly)" madness had apparently plagued the software testing industry in past years. And whether we want to admit it or not, that wave reduced a lot of testers to mere check-automators. A lot of testers I knew of switched to programming or other roles in the software field. Sigh!

Now that AI is unlocking a lot of possibilities around "all things algorhtyms", automating output checks may not remain as demanding task (and as daunting in some cases) as it has been so far. At the same time, applications based on AI generated code and new complexities demand thorough testing with context appropriate test strategy.

It simply means testers need to be more intelligent, smarter than their AI counterparts if they are to effectively collaborate with the new tech. To ensure that AI outcomes are reliable, the tester needs to think critically, while ensuring that they are not getting manipulated or tricked by AI. With increased complexities come bigger risks. And the role of these risk finders (call them testers or whatever) has now become more important than before. Only those who can think critically, those who can explore, observe and analyse deeper, those who can draw meaningful inferences and correct conclusions will be more sought after than those who don't.

It's high time to think about testing as something beyond output checking and also a time to test our thinking critically.

As usual, "Tea-time with Testers" will continue to share time-critical and relevant knowledge shared by passionate testers/experts in the community. And if you have interesting stories to share, we are all ears.

Until next time!

Sincerely,

Lalit

LALITKUMAR BHAMARE
CEO, Chief Editor "Tea-time with Testers"
–
Manager - Accenture Song, Germany
Group Lead - Innovation & Thought Leadership, Accenture QES DACH
Director - Association for Software Testing
International Keynote speaker.
Award-winning engineering leader.
Software Testing/Quality Coach.

Connect on Twitter @Lalitbhamare or on LinkedIn

WE'RE NOT DEAD! – THE ART OF EFFECTIVE TEST LEADERSHIP

- SIMON PRIOR

Simon is heading up the Quality Engineering teams at EasyJet. In his career he has worked in various roles across IT from C++ developer, Scrum Master, Build Engineer before finally finding his passion in Testing. He's worked in various domains from CyberSecurity, Gaming/ Gambling and now the Airline industry.

He has a keen interest in finding ways to improve the testing process and ensuring products are released with the highest possible quality. He also loves coaching/mentoring his team and others to be the best they can, enabling them to use all possible tools and skills to get their job done to the best of their ability. He has recently become a mentor for Leadership and Software testing on MentorCruise.com, check out his services here: https://mentorcruise.com/mentor/simonprior/

Simon is a well known speaker and meetup organiser in the Software Testing world, a Podcast co-host (check out TestingPeers.com) and also a keen advocate for companies to consider Neurodiversity as part of their inclusion programmes.

Simon was recently awarded Testing Manager of the Year at the European Software Testing Awards 2021, with a mention to his great people management and contributions to the wider testing community.

~People

The Tech industry is evolving faster than anyone is prepared for and with that, roles within the software world are in desperate need to change too, no more so than in the Testing world. With the demand for faster delivery, more complex tech stacks and more fluid ways of working, Testing roles and activities need to move too.

Test Managers as they are currently

One role which has been particularly highlighted in the trend of "traditional roles are dying" is that of the Test Manager. Taken from a Techwell article from 2018, this quote sums up the traditional test manager quite well –

In traditional software processes, test managers are responsible for all management aspects of their team. They dole out tasks and assignments, hold frequent meetings to stay on top of progress, review and approve estimates, and often provide technical guidance as well.

This traditional TM role is no longer effective and maybe not even relevant in a lot of companies (not all!), so there is a need to evolve the role. This is coming from a passionate Test Manager who wants to remain relevant and move both myself and my teams forward.

A siloed Test team who are only involved actively towards the end of a project is no longer the norm, so a TM being the only contact to the test team until the test phase is no longer enough. In some environments, it was the TM who provides that early estimate of the test effort on behalf of the team and will then hold their own team accountable to this value, they decide all resourcing, devise plans and communicate progress to the wider business. Their team left in the shadows until something goes wrong during testing or a defect leaks into production at which point the team are thrust into the spotlight.

The knowledge and accountability for testing and quality should no longer sit on the shoulders of one TM. The role of signing off on the quality of a product should not be the sole responsibility of the TM or even the test team and don't get me started on the term "gatekeepers of quality".

The Change to Test Leadership

While in some (if not most scenarios), the title of Test Manager won't change (at least straight away), the underlying role needs to evolve and be fundamentally different. The role ultimately comes down to the following priorities:

1. Building an Effective Team

Identifying what your team needs to be, starts with understanding whether you are the type of leader your team would want to work for. Are your team able to do their best work? Are they set up for success? Are you all working towards the same goal? Do they believe in your testing values?

2. Nurturing, coaching and empowering the team –

Regular face time with your team should be an important part of your time in the office. It's not necessary to have big team meetings every day, but being around for 1-1s (ensuring they aren't pushed out or cancelled regularly) where the focus is on them and their personal development and not project status. Give your team the confidence to feel they are making decisions on your behalf. This only becomes natural to them if you have worked closely with them to ensure you are on the same page and you've earned your trust from your team for them to know you will support them. Coaching and mentoring your team on testing and how to effectively communicate to the wider delivery team should be a key part of your discussions.

3. Being an advocate of your team and also of Testing best practices

You should be your teams protection and biggest cheerleader. When their leader is taking a collaborative decision which they have been involved in making to higher levels or other teams, the teams confidence will boom. Equally, when something does go wrong, protecting the team from the immediate backlash and then working with the team constructively to resolve an issue with an attitude of "we're in this together" will go a long way to preserve the teams confidence and trust in you.

Also, the other part of your advocacy should be to promote testing best practices to the wider business, find opportunities to get the TestSphere cards out with other areas and discuss all the great work your team and the wider delivery teams are doing to ensure high quality will help raise the profile of your teams work and make everyone aware of what testing involves.

It really is important that the team feel you are on the same journey with them and not focussed on your own agenda. Yes you are their manager, but manager doesn't mean you are above them. Your team are your biggest asset and working with them will make them more valuable

4. Being a servant leader

Management can be a thankless task, but being there for your team has to be your primary objective. Unblocking them so that they can excel should be one of your primary objectives. "How can I help you?" should be a common phrase your team hears from you, and you should always be working with them to improve their testing ability, empowering them to unblock themselves.

The primary goal of an effective leader is something one of the best managers I ever had passed on to me:

PEOPLE over PROJECTS

Ultimately, this means that your primary focus should be building, enabling and supporting your team instead of being "at the helm" and being the sole decision maker for all project work. If your team feel you have their back, they will feel empowered to make decisions that you would agree with.

That's All Very Nice But What About Getting the Work Done?

In the modern agile or devops ways of working, it is very unlikely that if you have more than a few people, that they will all be working on the same project or the same features, so there may be a need for someone to be overseeing all progress across potentially many streams. This may not be a Test Manager, it may be an engineering manager or some form of delivery manager where dev/test etc is all combined. But there will still need to be someone who is the Test/Quality champion who can help improve the processes from a test perspective across the teams.

Ultimately, this will require you and your team to devise a process where it is easy for you to keep on top of all the moving parts. This may include devising dashboards which highlight progress and testing pain points for you to assist with. It may also mean ensuring you have the right metrics in place for real time testing progress to be quick to access.

Ensuring testing involvement during the whole project, advocating for Testability and other ways to build quality in earlier should be a focus of the whole team, but having someone driving this from a leadership perspective will help move the team forward.

Can I Still Be A Test Manager?

You can, but the role is evolving. The focus should now on your team and raising the awareness of Quality Engineering/Testing, rather than leaving it as a siloed activity and you being 100% accountable for the quality of the product.

HUMAN PROTOTYPES OF QUALITY

- JOEL DOAT

Joël Doat currently works as a freelancer in the fields of software quality, technical communication, and teaching. He has a background in mathematics and is currently studying philosophy. His interest lies in conceptualizing software development and our relationship with the digital from a mathematical-philosophical perspective.

LinkedIn- https://www.linkedin.com/in/jo%C3%ABl-d/

The Cyberphilosopher: From Quality Assurance to Quality Wisdom

In a world where technology's tentacles reach into every corner of our existence, it opposes the old singularity which was only used by a small number of people. It is an all-encompassing phenomenon ingrained in every aspect of our lives. Gone are the days when technology could ignore the intricacies of society and culture.

Entwined in this complex web, such socio-cultural and anthropological influences come with the burden of ethical dilemmas in epic proportions. With my eyes glued to a blog post by Charlie Gedoen, the vision of "the rise of the technophilosopher" triggered my descent into QA's role of carrying this philosophical load; a quest, only for the bravest of engineers and testers, into the digitally soaked abyss that awaits us.

In his realm of "technophilosophers", philosophy isn't just a mere accessory; instead using it like self-made scalpels, rigorously dissecting data like a mad scientist's experiment. He cooks up ethical business logic spiced with hints of morality and societal norms; a stew pushing the boundaries of what we understand as software quality. Statistical models, insights at the depths of the minds, and there I am, nodding in agreement to "philosophical data scientists"—however, not the only facet of the mad coin; the other one being the savage in the philosophy of technology.

Let's look at how Alex Scott defines it:

> *"Technophilosophy (or technological philosophy) may be defined in a variety of ways. It may include the philosophy of technology ("tech philosophy"), the philosophy of engineering, the philosophy of computer science, and cyberphilosophy, as well as technofeminism, technocultural futurism (including Afrofuturism, Latin@futurism, and other futurisms), and other cultural or aesthetic movements embracing the philosophy of technology, the philosophy of science, and social philosophy."*
>
> (Alex Scott, What is Technophilosophy?)

Cyberphilosophy, a hybrid between digital tech and humanities, a potential game-changer in the making, is not solely being your run-of-the-mill academic discipline. In practice, wondering what sets it apart? It's a two-way street; not just crunching numbers enlightening philosophical puzzling, but hurling puzzles at the barricades of technology, all while keeping a squinted eye on the context – indeed a recipe for progress.

Picture this: dissecting software requirements through the prism of language, cryptic messages of stakeholders, expectations like wildfire, obvious question – does the end product line up with such a linguistic daydream? In QA, you have fancy tools, notations, but no guarantees that communication is in sync with the coded reality – truly a coding enigma. See, humans, they're not robots; they've got quirks, complexities that defy the neatness of the perfected notation system. Philosophy, my friends, is the off-road vehicle for the rough terrains filling such gaps while rattling our cage, and challenging assumptions we mindlessly swallow every day.

You might've picked up on it by now – that last example was a breadcrumb trail towards the heart of the matter: a quest of guaranteeing expected functionalities is what this quality assurance and software testing game is all about; forever on the hunt, seeking the right incantation to validate the ever-shifting situation before us. But don't be fooled by the path so far, it's not just philosophy that lights the way. In this age of hyper-specialization, we often overlook the hidden treasure in interdisciplinary synergies.

Now, here's where this twisted road for testers gets even wilder. Staring into the crystal ball, pondering the future of their trade, realizing it's not just about testing anymore; we've got these newfangled titles like "SDET" (Software Development Engineer in Testing) increasingly rising in the digital ether. Even more so, beneath it, still lurking in the shadow, AI like GPT-3 conjures test cases as if by dark magic, casting a dark cloud over the future of testing. So, where does that path lead us, you might ask? At first glance, it seems like a one-way trip into the unknown. But let me tell you, while the technical wizardry of QA roles is getting snatched away by cunning developers or soulless AI, don't despair, there's a silver lining. It can set us free, liberating us from the prison of mere rule-based applications. Unshackled from algorithms, we can wander free as true human guides again which this digital world so desperately needs; it's the tale of the cyberphilosopher teaching us the transition from quality assurance to the realm of quality wisdom.

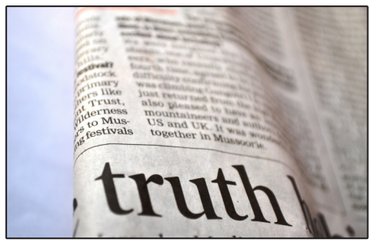
So, what's the real point here, beyond all my provocation and dramatization? Let me introduce you to a fresh concept, straight from the mind of Anne-Marie Charret—the Quality Coach. More than just testing, it's pushing the boundaries of tried-and-true quality approaches. Not restricted by the technical, the quality coach isn't confined to being a bug hunter; they're out there consulting, mentoring, marketing, and advocating for a new order. In a nutshell, it's a prime example of how the quality assurance mindset is evolving, or at least how the community hungers for change. And in these territories, you'll find soft skills often preached in the humanities, especially in the domain of "advocacy", explaining the term "quality" beyond its surface. Peeling back the layers of mundane pro-con lists and black-box tactics, it's about delving deep and digging up what lies behind the façade beyond the naively accepted truths.

Sources:

·https://charliegedeon.medium.com/rise-of-the-technophilosopher-a3cbdc77ca88

·http://philosophyreaders.blogspot.com/2018/09/what-is-technophilosophy.html

·https://www.annemariecharrett.com/quality-coach-explainer-to-cto/



Tea and Testing with Jerry Weinberg

JERRY WEINBERG
October 27, 1933 – August 7, 2018
–
Gerald Marvin (Jerry) Weinberg was an American computer scientist, author and teacher of the psychology and anthropology of computer software development.
For more than 50 years, he worked on transforming software organizations. He is author or co-author of many articles and books, including The Psychology of Computer Programming.

His books cover all phases of the software life-cycle. They include Exploring Requirements, Rethinking Systems Analysis and Design, The Handbook of Walkthroughs, Design.
In 1993 he was the Winner of the J.-D. Warnier Prize for Excellence in Information Sciences, the 2000 Winner of The Stevens Award for Contributions to Software Engineering, and the 2010 SoftwareTest Professionals first annual Luminary Award.

For over eight years, Jerry authored a dedicated column in Tea-time with Testers under the name "Tea and Testing with Jerry Weinberg". As a tribute to Jerry and to benefit next generation of testers with his work, we are re-starting his column.

To know more about Jerry and his work, please visit his official website http://geraldmweinberg.com/

The Technology of Human Behavior - Part 3

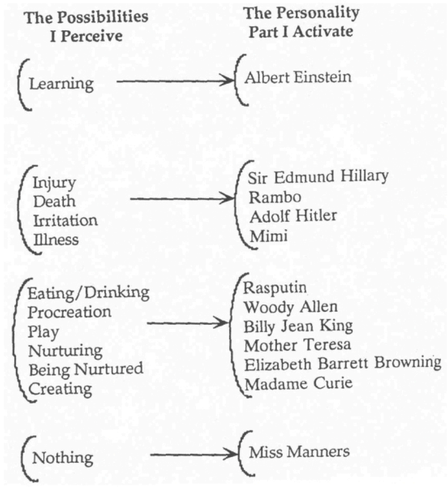


Figure above shows how these parts are activated by the possible consequences I perceived in Figure 11-5. Each part then produces a characteristic style, or personality, for handling each possibility. Some people who interact with me see only one of these parts, and they would be surprised if I should suddenly bring out another. Someone who knows only my Albert Einstein part would be shocked to see my Adolph Hitler part swatting flies, or dismissing a commodities salesman on the phone.

Yet both Adolph and Albert are parts of the whole me, and if you work with me long enough, you'll see both of them as well as many of the rest of the cast.

How meaning is developed?

Where do all these meanings come from to drive my interactions? Underlying the Satir interaction model we can imagine a tree whose branches lead to the various meanings Imay make in a given situation.

The root of the tree is my self—the essential part, the way I connect with the world around me. This is what some religions call the soul, or the inner light, or the spirit. Sometimes we meet people who don't seem to have one of these, but that's because it's buried under the higher branches of the tree.

The main trunk is formed by my yearnings—the natural or innate energy that translates my self into types of action. Some common yearnings are

- to propagate the species
- to survive as an individual
- to be loved
- to be taken care of, and nurtured
- to be healthy
- to be in harmony with nature
- to be valued
- to be whole

From each yearning branches one or more expectations—the translation of universal yearnings into specific ideas about how the world works to produce or withhold

what is yearned for. For example, different people wanting to be valued may try to satisfy this yearning in very different ways:

- to work very hard and do exactly what they are told
- to entertain everybody with clownish antics
- to produce and rear many babies
- to think of things nobody ever thought of before
- to get lots of money

Expectations are often held in the form of rules, such as one of mine that drives me to write books:

If I think of things nobody ever thought of before, I will be valued.

The rule expresses the way I expect the world to work.

The expectations combine with perceptions as well as my cultural background to produce meanings . Suppose I think of lots of things nobody ever thought of

before, but you tell me you don't like what I've done. I may interpret your feedback to mean:

- You don't value me, so I must not have thought of enough things.
- You don't value me, so the things must not have been original.
- You are a bad person, who doesn't play by the rules.

To be continued in next issue...

INTERVIEW

Greetings, TTwT readers! For this month's interview, please join me for a cup of tea with Janet Gregory, whom many of you may know from her and Lisa Crispin's best-selling books Agile Testing and More Agile Testing.

Hi Janet! I looked on your website Agile Testing Fellow by Janet Gregory and Lisa Crispin and I noticed you have a pretty busy schedule ahead of you. Thank you for taking the time to share your thoughts with the TTwT community.

What has changed for Agile and Testing? What is holistic testing and why is it important? Hear from Janet Gregory over a cup of tea.

- INTERVIEWED BY DAVE LEVITT

ISSUE 01/2024
OVER A CUP OF TEA WITH JANET GREGORY

Q1: One of the things I like about facilitating these interviews is the preparation work I need to do ahead of time. It's time consuming, but fun and a great learning experience, and in this case, quite reflective. I could not help but notice that your last book "More Agile Testing" was published in 2015. I was blown away that that was almost 10 years ago! Besides feeling a little older (but hopefully wiser), I think this calls out that software testing is just as dynamic as programming or any other discipline. Yet, like most practices, there are fundamentals that don't change. What do you think are the core fundamentals that haven't changed over these 10 years, and why?

Funny that we (Lisa and I) were contemplating how old our books were – Agile Testing (our first one was published in 2009 – almost 15 years ago. We wrote our September newsletter looking back at that book and what we would say differently. There are small things like more tools available, but the basics – the fundamentals are all still the same. Collaboration early in the process trying to prevent misunderstandings which are often the cause of defects in the code. Giving feedback early and often. Making what you do visible – draw out your problems, share with others to solve them. Think whole team! The second book – More Agile Testing goes into practices like acceptance-driven development and greater depth into different contexts such as regulatory, or embedded systems, and we've included stories from real teams to show what is possible.

Our third book Agile Testing Condensed is 100 pages – a great overview for people new to the topic, or as we hope – something that managers can pick up and get a high-level understanding of the practices.

Q2: There has certainly been a lot of big changes in the software testing community since your last book. What do you think are some of the big ones, and why?

There have been changes in the testing community. Some good, some not so good. I believe in whole-team testing, but that doesn't translate into firing all the testers, which far too many organizations have done – again. When organizations do that, they leave the teams without the necessary skills to test well. Instead of firing the testers, change their role into more of a consultant type role, teaching the programmers good testing skills, so they can expand the testing breadth of the whole team. Researching new ideas, getting better at what they do. Too often I see the programmers struggling with basic testing skills because they've never been taught. Unfortunately, poor quality is hard to recover from.

Some good changes I see are the number of tools that are now available and many of them open source. I still view AI with some skepticism, but you can use ChatGPT to generate ideas, and then vet them to your context.

One thing I am quite proud of is the Holistic Testing model. I used Dan Asby's Continuous testing model (or as I like to call it "we test here and here and…" model) as inspiration. You can find out all about it in my blogposts or better yet, download our mini-book Weave Quality into your Product from the AgileTestingFellow.com website. It provides a way for teams to talk about the delivery lifecycle and what types of testing can be done at the different times.

- JANET GREGORY

Janet is a testing and process consultant with DragonFire Inc. She specializes in showing agile teams how testing activities are necessary to develop good quality products. She works with teams to transition to agile development and has taught agile testing courses worldwide. She contributes articles to publications and enjoys sharing her experiences.

For more about Janet's work and her blog, visit https://janetgregory.ca or https://agiletester.ca

You can also follow her on twitter @janetgregoryca or LinkedIn.

Interlude for TTwT readers: the next few questions are based on Janet's and Lisa's excellent mini book on Holistic Testing. I highly recommend you download it from their website: Agile Testing Fellow by Janet Gregory and Lisa Crispin

Q3: There's no doubt that test automation has advanced leaps and bounds in the last decade. More on that later, but I am a huge advocate of old-fashioned static testing skills. To be effective and comprehensive, it needs to include two parts: what's there, and more importantly and much harder, what's missing. Asking these "what if" or "what about" type of questions is exploratory testing. However, it's traditionally been associated with testing the finished product, but as you call out, it's much broader in scope. It includes everything and everyone, which I see as one of the key points in your work on holistic testing. With that as a pretext, do you see many non-testers attending your seminars? If not, any ideas why?

In our courses, we encourage the whole team to attend, because it it the team that needs to fix any issues in their process. The very first time I taught the course, it was to a group of testers. They loved it, and had great things to say, but the next week I got a call from the leadership group who said … "the testers said all the programmers need to take the course too, or we can't change anything", so the next course was for all the programmers. I also taught a version of it to all the product owners. After that, I try to include at least one programmer, and the product owner in every course, but my preference is still for the whole team.

Often the best feedback is from the programmers because they learned the most. They had no idea how they could be helping with testing activities.

Q4: I am curious what led you and Lisa to write your mini book. Was it intended to be a natural extension of your earlier books, or was it something totally different?

For Agile Testing Condensed, we were responding to our readers. So many folks told us they wished that their managers would read our Agile Testing books, but it was too big. Lisa and I struggled to try to get our ideas. We kept it small in the hopes that more people would use it as an introduction into the ideas we expressed in the larger first books.

The mini-book, Weave Quality into your Product has a different reason. I had written a four-part blog post about the holistic testing model, and Lisa and I wrote several follow-up blog posts, but there wasn't one common place to refer people to. Hence the mini-book – combining all the ideas from the blog posts we'd written in one place. We decided to make it free so that people would openly share it.

Q5: I have a t-shirt that was provided by a testing vendor that says on it test.everything(). After reading holistic testing though, I think a better slogan would be something like everybody.testEverything(). Seriously though, the depth and breadth of holistic testing, along with the various technical and non-technical skills is mind boggling. As the old saying goes "Rome wasn't built in a day", how would you recommend a team or organization get started on this journey?

You bring up a question that many teams have. I think the first thing is to recognize the kinds of testing that might help your product. Most teams know about functional testing – making sure your feature does what it is supposed to do, and perhaps adding some boundary conditions to catch obvious issues. But there is so much deeper testing that can be done, and often there is not the time nor inclination to figure out what that might be.

To help teams discover what that means to them, I start them talking about risks. Product owners understand that. There are games like Risk Storming that can help teams consider diverse types of risks. Or Lena Nystrom's Would Heu-risk It cards to help think about different risks and heuristics. Once you know what your product risks are, it is much easier to talk about the types of tests that can mitigate those risks.

Q6: Extending on Q5: in your classes and seminars, I am curious what kind of reaction you get, if any, from testers who have been traditionally limited, either by training or by organizational norms, to testing just a finished product. To these people, holistic testing might be a radical change, and perhaps even viewed as a threat. Put another way, holistic testing needs an organizational commitment with a supportive environment to succeed. What do you do to entice senior leaders to attend your seminars and classes?

Unfortunately, there are a few folks who cannot see the possibilities. They have learned a specific way and believe that's the only way to do something, or they think it's not possible in their situation. I'm pretty sure we've all been there at one time or another, but it's hard to hear the line "But we've always done it that way."

I have been successful only once or twice in getting senior leadership to attend one of my courses, and even then, they came and went as they pleased. However, what I started doing instead was asking for a meeting with senior leadership at the end of a session. This enabled me to tell them my observations and give some informal feedback. Sometimes, I would follow up with a written report if requested to.

Q7: Obviously, I can't cover all the goodness of your mini book in this short interview, but I would like to talk a little about Dave Snowden's Cynefin model and the Chaotic Quadrant. This is the quadrant where people spend their time fighting fires, also known as unplanned work. It's important because no team can ever adapt any improvement, let alone one as broad-based a holistic testing, until this kind of work gets management visibility and put under control. From your experience, are you seeing this as a roadblock to adapting holistic testing? What, if anything, are you able to do about it?

I think that chaos is part of every team's experience occasionally. Things happen. How a team handles it usually depends on their experience and culture of organization. An experienced team may step back, look at why the chaos is happening, deal with it as quickly as possible, and try to figure out how to prevent it in the future. For example, if their production site goes down, it will be all hands-on deck until they fix the immediate issue. Then they will look at the underlying problem.

Where I see a bigger issue is when a team is constantly in chaos, fighting fires all the time. They have no time to improve. I once worked for a company who seemed to be in fire-fighting mode all the time. When I suggested to my manager that we needed to concentrate on fixing our process, he replied "But we are so good at addressing the fires." I quit shortly after that conversation because I knew the company culture thrived on the chaos. The company didn't last too many years after that.

Q8: I'd like to jump out of holistic testing now and jump into an automation topic that is approaching like a speeding locomotive. I'm talking about artificial intelligence and machine learning testing, and it's become the "controversy de jour". Have you any thoughts or opinions on this?

As I mentioned a bit earlier, I am still skeptical of the whole AI / ML bandwagon. I am sure there is a future for it, but right now, it has limitations. So much depends on how it was trained – the material used, where it comes from, how trustworthy it is, how many biases have been built in.

I see its biggest advantage right now, to help get someone started. It can generate test cases, but they may not be the best ones. They can generate code, but if the ML app learned on bad coding examples, it will give you bad code. It can give you great ideas for test data, but again, people need to use their critical thinking skills to identify whether it is truly representative.

ISSUE 01/2024
OVER A CUP OF TEA WITH JANET GREGORY

Q9: For my last question, I understand you are semi-retired now. I am as well, but I must wonder, in our field, is there really such a thing? It's always changing. Any parting words of wisdom or topics you'd like to share?

I laughed at this question. I'm not sure if there is such a thing as fully retired in our field. What I do know is that I have many other things I want to learn outside the field of software development. For example, I want to relearn how to draw, and get much better at my painting. That takes time. Something needs to be let go. It's been a hard decision, but one of the things I've let go is in-person consulting and conferences. My last conference that I plan on attending is Agile Testing Days in Potsdam, Germany in November 2023. Never say never, but I am feeling quite firm about that decision.

When I suggested to my manager that we needed to concentrate on fixing our process, he replied "But we are so good at addressing the fires." I quit shortly after that conversation because I knew the company culture thrived on the chaos. The company didn't last too many years after that.

Lastly, and kidding aside, whatever, or however you decide to spend your extra time, I hope you stay safe, happy, and healthy, and I thank you for all your time and insights for this interview!

Thank you for the opportunity.

~Processes

SOFTWARE TESTING: EFFECTIVE PROPOSAL WRITING AN ART OF WINNING BUSINESS

- DR. SANJAY GUPTA,

Dr. Sanjay Gupta received his Doctorate from the Indian Institute of Technology, Mumbai, (IIT Bombay) India. Currently he works as Software Delivery Director at NTT DATA. He is a Sun-certified Java programmer as well as Sun-certified trainer. He has published nearly thirty research papers in international journals and presented research papers in more than fifteen international and national conferences. His credentials have earned him a place in MARQUIS WHO'S WHO in Science and Engineering. His current areas of research, study, and knowledge dissemination are US Healthcare, Insurance, Software Testing, Talent Development, Java, Swings, J2EE technology, performance and Automation testing tools, Estimation Techniques etc.

He can be reached at skgiitb@yahoo.co.in

Abstract:

In the present competitive environment with changing technology and fluctuating economy, it is very important for a successful IT service organization to seize and win an opportunity which has many competitors. Quality of Project Proposal in terms of the accuracy in efforts and keeping customer first approach as a response to a bid plays a major role in winning the business. This article highlights important factors to be taken care while proposal writing solution.

Introduction:

In the current trend, selecting a service provider/vendor after floating a RFP (Request for Proposal) or against the requirements from an organization/customer depends on many factors like costing, technical & functional credibility, Trust, security considerations etc. It is important to know those factors which makes plays an important role to win a deal. Few of these are right costing, solution approach, futuristic technology view, cost optimization, automation and many more along with the recommendations from the existing customers.

Let us start from a point when the Sales Team comes up with information of new opportunity. This information can come in many ways like:

- RFP floated by the Customer

- A new requirement from a customer with whom you are already engaged from past or

- Information that a customer is looking for this engagement and let us give a proposal in a proactive manner.

In some cases with the ongoing engagements, we can easily find out that there are other areas where we are not currently engaged and can help our customer by providing the required services keeping their organization level vision in mind. If we can propose a solution/ service to the client in more efficient way to increase the quality/ productivity, it will be a good proactive initiative to get business from existing customer.

In many cases the turn around time to response to RFP varies from a week to months. One needs a very systematic and successful approach to meet the response time lines. Let us consider that a bulky document from the client (may be in couple of hundred pages overall) is received describing their requirements. Very first important tip is to go through the document at high level in couple of hours. This will give an overall idea about the need of the customer. After first round of the reading, solutioning team will have a good idea about the client business and requirements etc. This is the right time to document your understanding in a word document or a good suggestion will be to use a MS Excel spread sheet and create a work book for a given assignment. Separate worksheets tabs can be created toward distributing the complete understanding like:

1. General information about the client

2. Estimation basis, efforts,

3. Estimation assumptions,

4. Doubts, questions to be asked and Risk etc.

Now in the 2nd round of reading and discussion with the solutioning team, analyze the information in details and start filtering the important information in a spreadsheet. Once a detailed analysis is completed, a valuable workbook containing the entire key information with respect to requirements are handy with you.

In many cases, if the sufficient information from the customer is not known, it is important to gather the correct requirements. One way for this is to submit a Questionnaire with the customer which contains all the queries you need for sizing the project/application/system. It is always good to divide the questionnaire separately for Development, Functional & non-functional Testing or any other services required by the customer.

This article focuses on the software testing part of the Software Development Life Cycle (SDLC). In this case some important information to be gathered may be:

Client Synopsis: Primary business, Location, Contact details, Annual IT Budget, IT Team size, numbers of applications/system in scope etc.

Application Overview: Business domain, Technical Architecture, Brief description of the Applications/System/functionality in scope, new application development, Migration, Package implementation or customization etc.

Development overview: Internal developed or vendor has developed the same, Team size and their location, Technology used during the development and which model or methodology used during development like water fall, V- model, Incremental model or spiral model, Agile etc..

Existing Testing organization details: Internal or vender, Team size and location

Tools Details: Details about the tools for Development, Test Management, Change Management, Defect Management, Test Automation, Performance Testing, Configuration Management etc.

Process Maturity level of the customer: Some details about the organization maturity levels like CMM level, ISO etc.

Services required: What type of testing services customer is looking for like: Test Consultancy, Functional Testing, Test Automation, Performance Testing, User Acceptance Testing, Testing on Cloud etc.

Others: Information like Test Data availability, details about Test Environment, Skill required if anything special, project timelines etc.

Apart from the above information, specific to different testing types, one can also ask some questions specific to a particular service. For example, if the customer is looking only for Test Automation services, one should know the number of test scenarios, Test cases to be automated. A demo of the application/product will be a good idea to have before you start estimation.

In the similar lines if Performance Testing is one of the requirements some basic information described below will be help you to estimate the efforts:

1. Type of application like web application, client server etc.

2. Connectivity to the application like LAN, WAN,

3. Protocols used to connect the application from user machine like JDBC, ODBC, and HTTP etc.

4. Performance bottleneck in the application,

5. How many Users are using the current application? What is the expected numbers in near future?

6. What is the average and peak load of the users for the application under scope.

7. Active user details during peak and average hours.

8. Information about the key business functions that will be executed in the application by the end users.

9. Business function flow diagram for the application. if customer is looking for those services you would need the following information like:

10. Performance Testing environment, Test Data preparation details etc.

In the next section our main focus will be towards sizing/estimating the testing requirements. By now, after going through the requirement you will be clear about the Testing Services required by the customer.

These services may fall under:

- Test Management,

- Test Consulting,

- Functional Testing,

- Performance Testing

- Test Automation

- Other Specialized testing services like Games testing, Mission critical testing, Embedded System testing etc.

If the gathered requirements for the required services are correct, the estimations will be very accurate. The correct effort estimation is a win a win condition for both customer and service provider.

Remember that as a Service provider if you underestimate your efforts and win the deal- you are in Loss. On the other hand if you overestimate the efforts –you loose the bid. So coming up with a nearly 5-10 % plus minus efforts is an art to win a business and justifying your role. How to Estimate and size the application is the next most important task to know.

You will find that the gathered information may come in different ways like,

- Technology

- Lines of code

- Existing Test Cases

- Use Cases

- Function Points

- Technical Design diagram

- Business scenarios, Functionality of the application

- Number of Screens

- Number of Release cycle

- Number of regression cycles & regression automation %

- Test Automation coverage Vs total number of functional test cases

- Development & testing team size

- Hardware and software requirements, tools and many more.

The above information will help to calculate scientifically the efforts and some information like total number of the current development team size (in case if the customer want to outsource the testing activity to the vendor) will help to cross check the scientifically calculated efforts with the thumb rules across the software industry. For example a good ratio of 70/30 % between the development and test team is well accepted number for a new software development. On the other hand if it is a maintenance/ production support work a 15-20% testing team ratio to the development team is a good one to keep in mind.

If you have been given the line of code information with respect to different technology, there are standard tables like Capers Jones' language table which will help you to convert line of codes for a given technology into the function points. Now it is experience and your organization maturity level in Software testing which will help you in converting these function points to the Test Case numbers. For function point analysis you can go through any standard book or there is enough study material available on the internet.

In some cases you will find information about the business functionality of the application. It is advised to look in to end to end business scenarios and find out that how many steps are required to complete one scenario. On the basis of the number of steps one can categorize business scenarios as simple, medium and complex. . For example, if a business scenario takes 1-5 steps, this can be considered as simple. If it takes 5-10 steps or 10-20+ steps, it can be categorized as medium and complex respectively. Once you decide on the complexity and numbers of business scenarios, use the standard thumb rules for converting these scenarios into the number of Test Cases.

In the similar lines, if you have the use cases as input, try to analyze the same and find out how many business scenarios/test cases can be derived from them. When ever you reach to Test case level, try to break them in to the simplest one.

You can optimize your total number of Test Cases by using the well known techniques like Orthogonal Array to come up with the optimal set of Test Cases with maximum Test Coverage.

At this moment , solutioning team is very near to the effort estimation. Here are the few important factors to keep under consideration during effort estimation.

- System Study and Test Planning

- Automation feasibility Study/ (if Test Automation is a requirement)

- Project Management

- Knowledge Transfer phase

- Test Case/script Design

- Test Case Execution and many more

Depending on the testing requirements, test engineer efficiency, past project experience and organization norms, overall test efforts can be derived in Person Days, Person weeks or months. Once you know the number of the resources for a given time lines to complete the job, multiply it with the cost associated with different resource levels (like project/Test Manager, Project/Test Lead, Test or Automation Engineer) and arrive to the final costing.

If you are proposing a onsite/offshore engagement model, the travel cost for the resources to the client location and their accommodation should also be included in the overall costing. If you are going to use any software tools for a given engagement, don't forget to include the tools cost along with the number of required licenses.

I am attaching below a basic workbook (Technical Solution Summary book.xls) as a template for you to get a fair idea as an assignment. Take an old or the fresh proposal and start filling the data in it. The numbers/values filled in the sheet are just a number, they are not any standard values, and it is given just as an example. This worksheet will be useful for you during write-up, interacting with the customer during telephonic/personal interactions.

Essential Parts of the proposal:

Important sections need to put in the proposal are:

- Client Information: Give a brief description about the customer
- Your Understanding of the requirements: Describe your understanding in details
- Scope /out of scope : Mention clearly what is in scope and what is out of scope
- Your approach to complete this task
- Description of the services you are going to provide to customer
    - This should not be the generic definitions, but with respect to the given application and scenarios
- Effort estimation and resource plan with respect to the given time lines
- Deliverables
- Assumptions
    - Pen down all the assumptions you have considered during effort estimations.
- References
    - Write all the documents/ material names which you have received as part of RFP or the request considered during the effort estimation
- Appendix
    - You can include the important information about solution designer and your organization
    - Solution provider capabilities in terms of the case studies in the similar type of projects.

Apart from this the proposal writing template varies from organization to organization level. So gather all the information in a given proposal template. Get it reviewed with your peers/ practitioners and incorporate their opinion.

A customer is always looking for the support from the innovative solutions/ Processes/ tools/your organization has developed and utilization them as part of the services you are recommending.

At this point, you have a great Proposal with you, ready to submit to the client.

Summary:

Proposal writing is an art comes from your experience. This is the result of best practices derived from the success and failures. Thumb rules across industries may differ slightly. In this article, these are authors personal views for proposal writing. Detailed discussion in this article, guidelines, your thoughts, thumb rules, Industry standards will definitely help in estimating the requirements and writing a successful project proposal.

References:

“Accelerate Software Testing with Orthogonal Thinking” by Dr. Sanjay Gupta ; www.stickyminds.com ;

DATA MIGRATION TESTING

AND

TOP THREE HABITS OF AN EFFECTIVE TESTER

On December 18th, 2015, I signed the Personal Commitment for 'The 7 Habits of Highly Effective People', a signature program by FranklinCovey® . Thanks to FIS Global, my employer, for that free-of-cost training for their upcoming leaders. The top three habits on that commitment have to do much with this experience report I am sharing. But before I go further, I need to make three disclaimers.

1. I have been in data migration testing for around seven (non-linear and discrete) years.
2. I am a tool-agnostic data, database, and ELT (or ETL) tester who can plan, strategize, and do hands-on data migration testing w/or w/o a QA framework.
3. John Morris's four golden rules, defined in his seminal work scripted in his book 'Practical Data Migration,' influenced my latest thinking about data migration testing.
    - Golden Rule 1: Data migration is a business not a technical issue.
    - Golden Rule 2: The Business knows best.
    - Golden Rule 3: No organization needs, wants or will pay for perfect quality data.
    - Golden Rule 4: If you can't count it, it doesn't count.

So, after my disclaimers, let's look into

1. What, Why, Who, and How of Data Migration Testing
2. Top three habits to become an effective and efficient data migration tester.

Definition - What is Data Migration?

In software quality and testing space, trustworthy sources of information and data are paramount, which also applies to the definitions. Hence, I chose to pick one of the definitions of Data Migration from Microsoft®, which says.

In general, data migration means moving digital information. Transferring that information to a different location, file format, environment, storage system, database, datacenter, or application all fit within the definition of data migration.

Text source for the above mind map image: https://bit.ly/DataMigration_Ms_Def

To be more specific

Data migration is the process of selecting, preparing, extracting, and transforming data and permanently transferring it from one computer storage system to another.

A Case Study - Why do we need Data Migration?

There could be many reasons, and so far, I have encountered the following reasons for data migration. The reasons are listed in the decreasing order of # of data migration testing assignments my teams and I handled.

1. On-premise to cloud movement of software products and data.
2. Systems integrations (Transactional, Analytics, and Reporting).
3. Data-driven product maintenance, upgrades, and compliance.
4. End of life (EoL) for a specific data/server technology(ies).
5. Managing license cost(s).

The context of data migration will vary from one organization to another. I recently worked on data migration testing assignments in EoL (End of Life) context for three of our highly prestigious and Fortune 500 clients in Insurance and Retail businesses. You can read more here for such an EoL scenario.

To us, End of Life is an interesting case because data migrations in such cases often present testing challenges and learning opportunities originating from

1. Vendor(s) selection
2. Technology & version(s) determination
3. Identifying non-negotiable compliance(s)
4. Meeting strict timelines
5. Testing to meet seamless user experience expectations
6. and the most critical one is 'No Looking back,' simply because it is the End of Life!

So, once the migration context is known, our job is to identify who matters and what matters to them

Data migration - Who matters?

Customers and Users

The Customer (s): From my view of the business world, I see a dollar-paying client and a dollar-consuming 'us.' Both are concerned with Robustness, Efficiency, Reliability, and Accountability of the overall data migration planning, execution, and end results from the Inception to the Switchover date.

The User(s): Anyone, including customers, affected (in a good or bad way) by the migrated data. Period! More specifically, the target (internal or external or both) end-users of the WAN and/or LAN-enabled web and mobile technology products developed or maintained by our Customer's IT teams.

Learning: All customers are demanding, and some are helpful too. Some of the Customer and user representatives I got opportunities to communicate with were highly business and tech-savvy. They knew what they wanted to see in action and helped me gain insights into what they wanted from the data migration and testing team(s), and this made our job slightly more manageable.

Data migration - What matters?

One of our clients said humorously, **'I don't want the users to know if something changed except that they experience improved efficiency in day-to-day usage of the products!'**

Almost 90% of the customer representatives we interacted with care a lot about seamless user experience and no lousy impact regression, especially on data quality and operations. Sometimes, the conversations give me the impression that 'Data is the product itself, and it is no more a king but a more powerful queen nowadays.'

In a nutshell, at least from a client standpoint, a successful data migration seems all about consistent data quality, business operations continuity, and seamless user experience while using the existing products and systems.

Now, while the above statement can be considered too generic and not as a testable requirement, it can work as a great motivator in understanding the client's emotions and deep expectations about 'to be perceived' migrated data quality and UX.

Data migration Testing: In Practice

Are you smiling by looking at the image below?

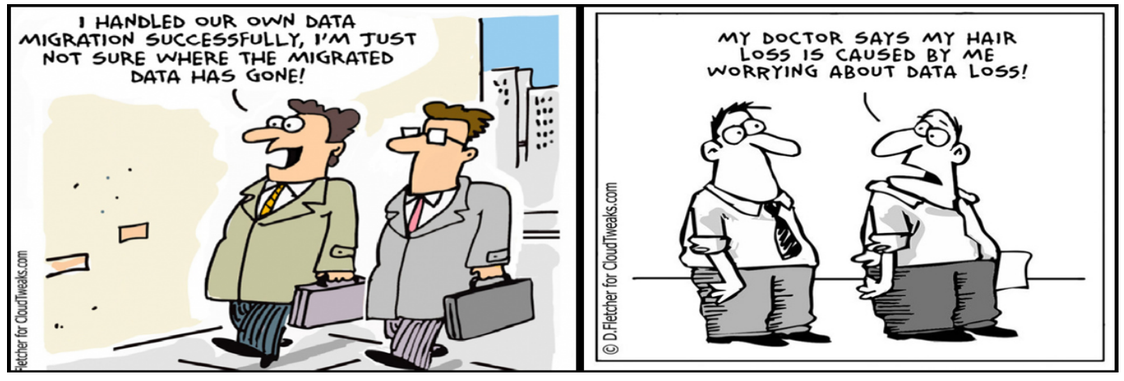


Image source: https://cloudtweaks.com/

When converting expectations on data quality into testable requirements, it is more practical and beneficial to think about risks associated with data migration. Historically, some risks are prevalent and directly associated with overall data quality and quantity. For example

1. Data loss
    1. Historical (or One time load)
    2. Incremental
2. Data duplicity
3. Data corruption
4. Upstream and downstream incompatibility
5. Data inconsistency

Some risks are associated with database(s), metadata, schemas, existing and new designs, data dependencies, etc.

It would be worth noting that while most data migration and data migration testing tools have evolved (or at least they claim) to successfully address the risks mentioned above to a reasonable extent, we (customers and us) have to be careful about testing migrated data quality beyond the tool vendor's promises.

My team handles data migration testing from a technology and technical perspective. We don't boast of thoroughly understanding the business, but we attempt to understand the business, products, and technical context of data migration. This understanding is an iterative undertaking and sometimes reaches towards the end of the data migration testing life cycle.

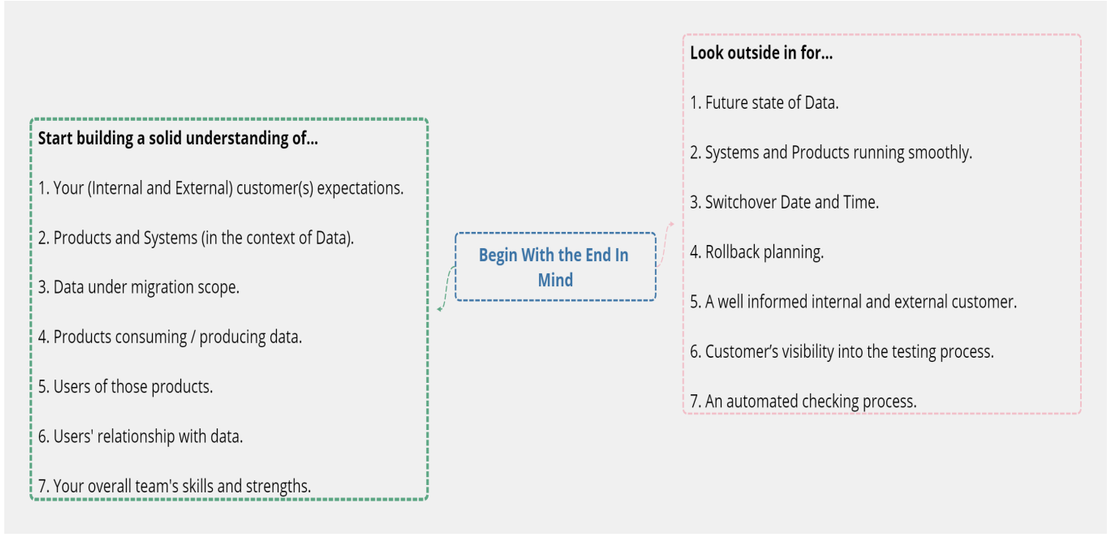
So when it comes to What, How, When, and Where of Data Migration Testing, this is where me and my team leveraged the top three habits.

- Be Proactive®
- Begin With the End in Mind®
- Put First Things First®

Habit #1: Begin With the End in Mind®

I prefer to start my test planning and strategy with Begin With the End in Mind® , which is also about setting mutual expectations and accountabilities. Personally, in the name of data, migration, or ETL testing, I hate to imagine preparing any test plan or strategy our customers can't understand. I think of making the plan, strategy, and all test ideas data the Customer's property at the end of the day and not making it Latin and Greek. Period!

Over time, I learned and tried to define the data migration and migration testing goals in two broader categories.



The data gathered and a shared understanding built with the help of the above ~14 essential items present in both categories helps in serving my habit #2.

Habit #2 Be Proactive®

As testers, we need to establish scope and determine priorities, business and product criticalities as far as transactions, analytics and reporting requirements are concerned, any system limitations, constraints, and possible pitfalls. We must do all these proactively and iteratively because you never know when business priorities change.

One of the critical lessons I learned was asking questions on

- What is the current state of quality of data?
- What are those day-one critical data requirements?
- What data can wait and up to what time and why?
- What Databases (DBs) are high priority, and what are lower?
- and a few more to start with.

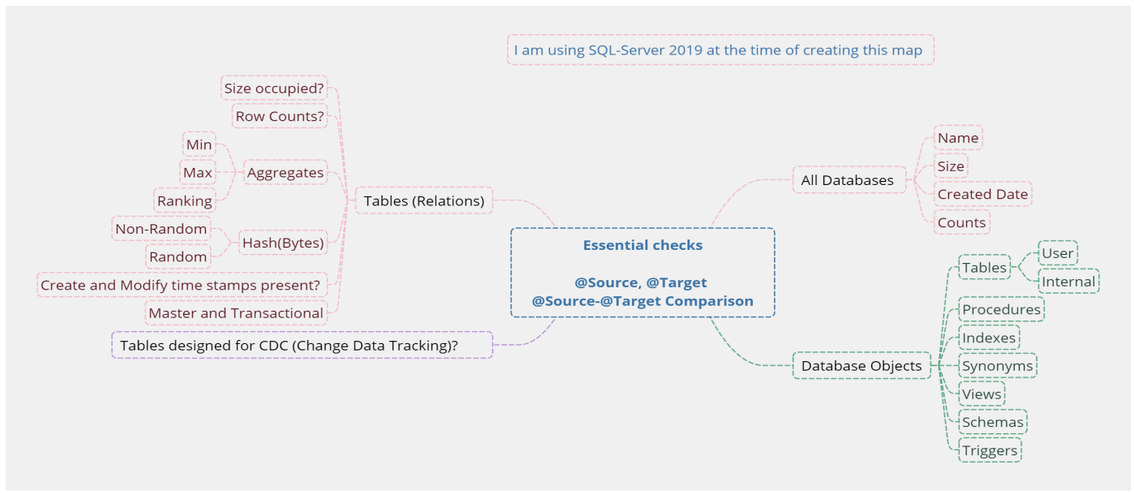
Keeping a vigilant eye and practicing the first two habits helped me to stay focused on the seemingly elementary but most time-consuming activity in the data migration testing space. I am now coming to mention Habit #3.

Habit #3: Put First Things First®

Putting first things first here indicates having a problem-solving mindset as far as test identification, script preparation and its automated execution, bringing efficiency and visibility into the testing process, making script execution easy for non-technical folks, and saving your valuable time for exploring deeper data problems.

In data migration-related tests execution, which essentially includes Extract, Load, and Transform, one of my goals is to automate the script execution part of testing.

The below image lists a bunch of essential checks that a data migration tester would want to run but with the use of standard SQL queries provided as a part of tool documentation and some automation (script, batch, schedule, and notifications)



***Note**: I used MS RDBMS MS SQLServer 2019 when writing these lines and creating mindmaps, and if you are from RDBMS. SQL Server and T-SQL background, you might relate to content more quickly than others.*

A glimpse of such to-be-automated portable script execution

```
PRINT ('Reusable script. RO access and DB State View permission is required on DB. If no access / permission raise a ticket')
PRINT ('Checking current state of Data and Database objects')

PRINT ('-- Query execution is started at --')
Select getdate();

PRINT ('Server' + ' ' + @@SERVERNAME + ' ' + ' Database ' + DB_NAME())

PRINT ('-- High level Details on All Databases : Name, Size, Created Date --')

EXEC sp_helpdb;

PRINT ('-- Counts and Details of DBs, Tables, Procedures, Views, Schemas and Indexes(At present) --')

PRINT ('Databases')
select count(*) as NumOfDBs from sys.databases
select * from sys.databases where name like '%Tribe%' order by name desc;

PRINT ('Tables')
select count(*) as NumOfTables from sys.tables;
select *  from sys.tables where type_desc = 'USER_TABLE';
select *  from sys.tables where type_desc = 'INTERNAL_TABLE';

PRINT ('Table Columns')
SELECT TABLE_NAME, COLUMN_NAME, COLUMNPROPERTY(OBJECT_ID(TABLE_SCHEMA + '.' + TABLE_NAME), COLUMN_NAME, 'ColumnID')
AS COLUMN_ID
FROM TheTestTribe.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'TheTestTribe_Events';
```

```
PRINT ('-- Counts check in All Tables --')
select count(*) NumofRows_TE from TheTestTribe_Events with (nolock);
select count(*) NumooRows_TC from TheTestTribe_ThriveCourses with (nolock);
select count(*) NumofRows_TV from TheTestTribe_Volunteers with (nolock);

PRINT ('-- HashBytes At Address level --')
select top 1000 HASHBYTES ('SHA2_256', address_line_1) from TheTestTribe_Offine_Events with (nolock)
where city = 'NOIDA'
and TheTestTribe_EventIDin (select top 1000 TheTestTribe_EventID from TheTestTribe_Events with (nolock)
where city = 'NOIDA' order by TheTestTribe_EventIDdesc)

select top 1000 HASHBYTES ('SHA2_256', address_line_1) from TheTestTribe_Events with (nolock)
where city = 'NOIDA'
and TheTestTribe_EventIDin (select top 1000 TheTestTribe_EventID from TheTestTribe_Events with (nolock)
where city = 'NOIDA' order by TheTestTribe_EventIDasc)

PRINT ('-- Aggregates for different fields --')
PRINT ('1. TheTestTribe_Events')

select min(EventID) as MinOf_TheTestTribe_EventID from TheTestTribe_Events;
select max(EventID) as Maxof_TheTestTribe_EventID from TheTestTribe_Events;

select city, count (distinct EventID) as CityWiseEvents
from TheTestTribe_Events
group by city
order by count (distinct EventID) desc;
```

```
PRINT ('-- Ranking --')
SELECT TTTE.city, count(distinct TTTE.EventID) NumofEventIDs
    ,DENSE_RANK() OVER
    (PARTITION BY TTTE.city ORDER BY TTTE.city DESC) AS Rank
FROM [dbo].[TheTestTribe_Events] AS TTTE
group by TTTE.city
ORDER BY count(distinct TTTE.EventID) desc;
GO


PRINT ('TheTestTribe_ThriveCourses')
Select min(CourseID) as MinOf_Thrive_CourseID from TheTestTribe_ThriveCourses;
select max(CourseID) as Maxof_Thrive_CourseID  from TheTestTribe_ThriveCourses;
select count(*)  from TheTestTribe_ThriveCourses;

PRINT ('-- The queries were stopped at below server timestamp --')
Select getdate();
```

I hope, by now, you have a fair understanding of

1. *What, Why, Who, and How of Data Migration Testing*
2. *Top three habits to become an effective and efficient data migration tester.*

Understanding, practicing, and refining the above two have worked well for me so far, and I hope this should work for you as well in the context.

- SANDEEP GARG

Sandeep is a student of software testing and quality leadership. He has over 17 years of Software / IT experience in different functions and is heading Bridgetree's QA/Testing unit, where he is accountable for overall test governance and hands-on testing of web/mobile apps, customer data platform, data migration, database, and ETL processes.

He has successfully built and led testing teams at major organizations such as Fiserv and FIS. Sandeep values context-driven testing principles and remains open to diverse perspectives and ideas.

Sandeep often speaks at local testing meetups, conducts workshops, writes on LinkedIn, and mentors and coaches software testing professionals.

ras Shypka on Unsplash

HEURISTICS, WEB DESIGN

VIP BOA – A Heuristic for Testing Responsive Web Apps

EDUCATION, SPEAKING TESTER'S MIND

ISTQB, Fever Dreams and Testing

EDUCATION, WOMEN IN TESTING

How To Read A Difficult Book

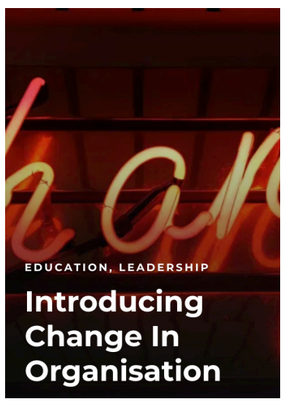
INTERVIEWS, OLD IS GOLD

Over A Cup Of Tea With Jerry Weinberg

mon Migaj on Unsplash

SPEAKING TESTER'S MIND

Software Testing And The Art Of Staying In Present



EDUCATION, LEADERSHIP

Introducing Change In Organisation

PEOPLE AND PROCESSES, WOMEN IN TESTING

Addressing The Risk Of Exploratory Testing Part 2

CAREER, PEOPLE

Managing Multiple Passions

LEADERSHIP, OLD IS GOLD

Leading Beyond The Scramble:

After nearly twenty years of working in software, I
many companies. One of them is what I call the sc

INFOCUS

Do you know all these amazing articles?

Great things survive the test of time.

Over the last ten years, Tea-time with Testers has published articles that did not only serve the purpose back then but are pretty much relevant even today.

With the launch of our brand new website, our team is working hard to bring all such articles back to surface and make them easily accessible for everyone.

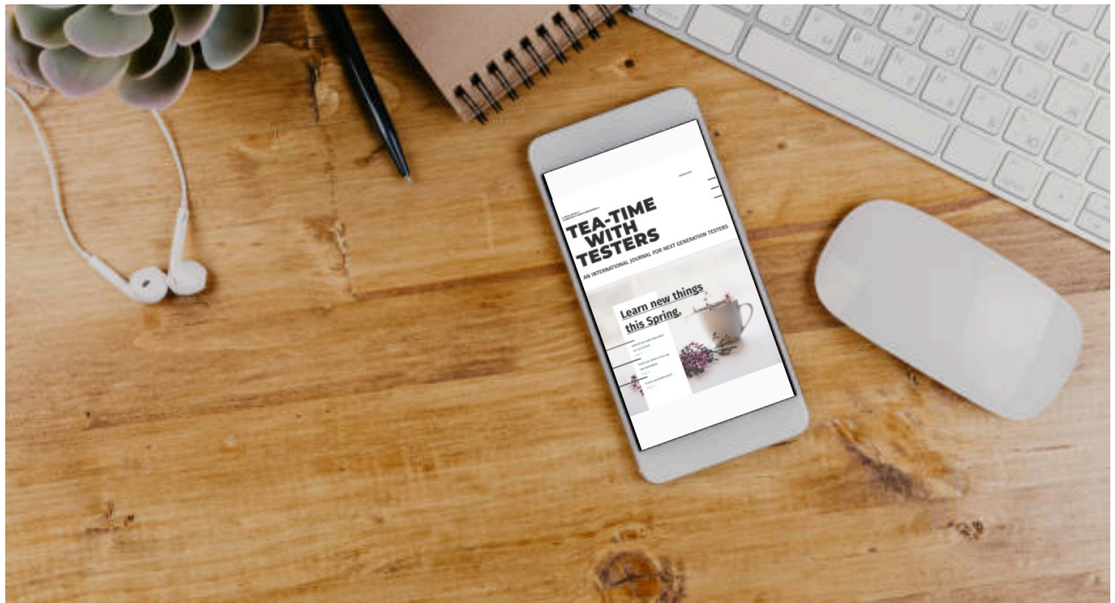
We plan to continue doing that for more articles, interviews and also for the recent issues we have published.

Visit our website www.teatimewithtesters.com and read these articles.

Let us know how are they helping you and even share with your friends and colleagues.

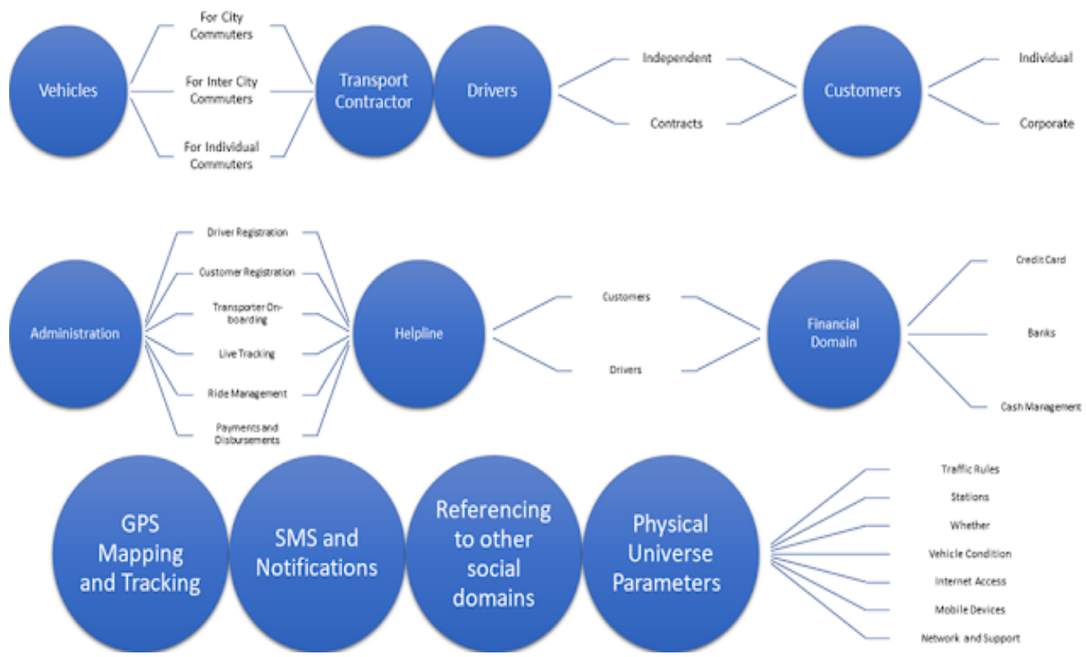
If you think we could add more articles from our previous editions, do not hesitate to let us know.

Enjoy the feast!



~Products

PHASE SPACE: THE IMPLICATIONS OF TIME



Time plays a vital role in determining various states of software applications with respect to their objects, data, and users. It is indispensable for testers to conceive “Initiation-Termination” logs of each entity before the execution of test cycles.

Time is also related to the speed and mass of the object. In software terms, the mass of a coded object may reflect its complexity, the data it processes, the interactions it exerts, and its experiences from other objects it reflects. Therefore, if we apply the analogy of phase space with respect to time on a software object, we may also be able to calculate the actual object life-cycle and chronology of system interactions for the object, as in phase space, a system can keep all its forms, and so does its objects.

On the other hand, speed relates to the volume of the data, the life cycle of the entity from one instance to another while waiting, suspending, and transforming from one state to another.

Considering the example system of a ride-sharing application, if the testers can determine the life cycles of “Ride”, “Commuter”, “Trip Data”, “Vehicle”, and “Tracking”, the exploration and creative testing can become a controlled activity.

Creative testing is a mix of "User and exploratory testing techniques”, where the test team uses heuristics in combination with established user Journeys to discover bugs that can only occur without using any sort of written test cases or requirement specifications, while only referring to designs and variable flows.

This will lead to elaborative mental models and maps to further create detailed test cases and use cases for sprint-level and regression testing activities, and even automation.

Time - Object Life Cycles (or) The Orbital Lifecycles:

1. We need to understand that each object in an application eco-system has its initiation and termination life cycle. This is also a foundation layer for determining the actual user journeys.
2. The resurrection or the effect of orbital lifecycles of these objects then depends on the time, interactions, and data each of these objects digests in their respective cycles.
3. By “Orbital Lifecycle,” we mean to state that the relationship of the object with other objects in phase space is not linear, in fact, it is circular, whereby its core nature the application object either moves and interacts in an infinite loop or it can modify and terminate itself as per code changes, requirements, context, and feature updates.
4. These cycles can experience wobbling from other orbital influences, commonly known as the “Ergoforce”. Such as, the life cycle of a trip is in the continuous wobbling effect of commuters and trip data. The driver's orbit is in a constant push-pull state while checking in and checking out passengers and being tracked by the interactive map plugins.

Time - Object Saturation

An application object can be saturated due to several factors, such as:

With the data:

Where the object, either by the limits, conditions, or, volume capacity, the object starts to lose its traction with other objects and behaves in accordance with the subsequent conditions or creates an anomaly within a logical flow. To test saturation errors, testers use boundary heuristics which helps them break patterns, use a test stream of data, and input extreme random actions on objects and functions.

With the instances

Where the developers or the users have called the object instances to a limit to its saturation and results in loss of its orbital movement, gravitational pull, or response to other objects.

Several factors come into play here, namely poor internet connection, server calls, user behavior, and dead clicks. Instances rely on existence, and the fact is proved alongside frequent time intervals where we can predict the state of the object at a certain point within the application lifecycle. If we say orbital movement, we mean the regular lifecycle of an object, which is coded, well-defined, and has inputs, outputs, data, interface, structural, and platform support. Therefore, any loss/anomaly occurring within this lifecycle will result in abnormal orbital movement of the object.

Performance management

Where the objects fail to respond due to transaction, interaction, and even data loss in simple requests. For example, a bad internet connection on a driver's phone will result in a loss of vehicle tracking on the commuter's application. This will cause the system at the admin level to mark the ride as canceled or can invoke a series of SOPs in order to deal with such performance issues.

The Test Execution

Where the testers can fall into a simple trap of neglecting the results of their own findings due to the multiple executions of the same test cycles on an exhaustive level. What happens here, is when a tester keeps executing the same test cycle, will eventually change the test case flow, even if it works fine, and will fall into a pitfall of overconfidence.

The best way to avoid this is for testers to break the execution into several working sessions and then manage test executions on the basis of what's essential and what's not.

ARSLAN ALI - PRESALES AND SOLUTIONS CONSULTANT

Arslan Ali is a veteran software tester and solutions consultant from Pakistan. Having more than 20 years of diverse experience in information technology services and IT education, Arslan's main focus now is community building and consulting.

He is the founder of the BeingTester initiative and also co-founder of the Pakistan Software Testers Society and SQAs from Pakistan, where testers from around the country share experiences, assist other testers and help build a stronger QA community.

FAIZA YOUSUF - PRODUCT MANAGER AND COMMUNITY BUILDER

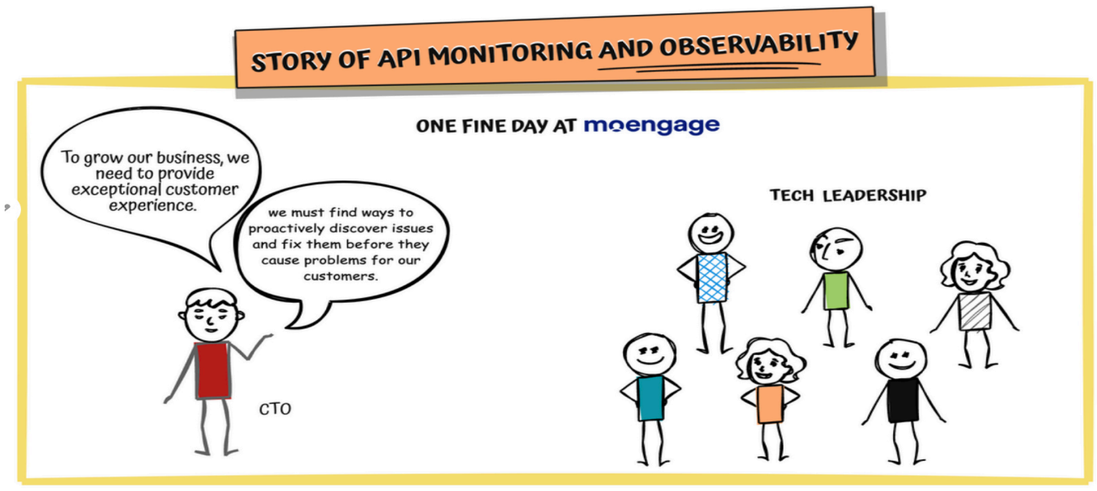
Faiza Yousuf is a software engineer with 13 years of experience building products and teams. She is a serial founder focusing on women's financial inclusion initiatives and improving gender parity in tech. Faiza is a multi-award-winning community leader. She founded WomenInTechPK and cofounded CodeGirls and CaterpillHERs.

Faiza is an Expert-Vetted freelancer on Upwork and was featured in many of their campaigns. She is a well-known international speaker, loves to read, and enjoys writing about tools and practices.

THE STORY OF API MONITORING

FROM GLITCHES TO GLORY!

Our CTO shared an inspiring vision with the company leaders - “To grow our business, we need to provide exceptional customer experience. To do that we must proactively discover issues in production and fix them before they cause problems for our customers”



A brainstorming session was held between the leaders to discuss how to turn this vision into a reality. Leadership realized that It is really hard to predict how an application can fail, and it can fail in new and creative ways. Not all issues are code issues, It is not always the erroneous code that causes problems for the customers. Incorrect configurations, software upgrades, human errors during deployment, cloud outages, scaling issues, etc can all cause problems.

Ideas started to flow! As we progressed, we arrived at a point where a question arose from our architects -

“How about we identify issues before our customer by running our automated tests frequently”?

Find bugs with automation? Are you kidding me?

To be continued...

INTRODUCTION

APIs power today's businesses. APIs are becoming an increasingly important part of modern software development. Nearly every digital interaction relies on APIs today. As software systems become more complex, microservices become more prevalent. Over the past few years, monolithic applications have been replaced by microservices that communicate through APIs. API is the building block of any modern application.

The ultimate goal of businesses is to provide a positive experience to their customers.

A downtime or API error will lead to support tickets and uninstalls from your customers. Furthermore, it adversely impacts your user experience, and in the long run, hinders your business' growth. Monitoring APIs therefore becomes crucial to offering your customers a flawless experience.

Customers are the top priority for businesses, and observability helps them accomplish that goal. Testing teams are increasingly taking responsibility for monitoring and contributing to overall observability initiatives. There is no reason why you as a tester can't do that too. To get started, you don't need to be an expert in DevOps. We're going to discuss API monitoring and an effective strategy to implement it. In addition, we will learn how it fits into the world of observability.

THE BEGINNING: THE IDEA OF API MONITORING TO ENABLE OBSERVABILITY.

Continuing the conversation...

Architects: "How about we identify issues before our customers by running our automated tests frequently"?

Me: ????

<In my mind>Find bugs with automation? Are you kidding me?

< Pauses >

<with some hesitation> Not sure of that. Let's experiment anyway!

I looked at automated checks as a way to eliminate repetitive work, and not as a way to identify problems. Although automated checks can detect regression bugs, I was not confident that they could serve as a way to detect new issues in production before the customer. I observed that some of our testers found more bugs when writing the automated checks than when they executed them. My initial thoughts were that it wouldn't work, but I decided to give it a shot. I did not want to dismiss this idea without experimenting. The more I discussed it, the clearer it became. Our goal was to identify issues and fix them before they impact customers. The plan was to use existing automated checks for monitoring and adding new checks in places where they were lacking.

To monitor the health of the application, we must run the automated checks frequently, and the tests must be fast, reliable, and intelligently designed. API automated checks are exactly what can accomplish this. They are faster, cheaper, and more reliable compared to UI checks. They can be run frequently to provide rapid feedback, making them an ideal candidate for monitoring.

WHAT IS API MONITORING?

API monitoring refers to the practice of running automated API checks frequently in production, to gain visibility into availability, performance, and correctness.

WHY DO YOU NEED TO MONITOR YOUR APIS?

- As APIs become a more integral part of customer journeys, monitoring them becomes more necessary.

- Today, the engineering teams deploy code changes several times a day to several times a week.

- Using API monitoring, teams can find errors and performance degradations as soon as they occur, so they can deploy a fix before it impacts customers.

- API monitoring will result in less downtime, fewer support tickets, and faster resolution times. Moreover, it will allow you to identify issues before your customers do.

Me: We are already monitoring our service. Don't we have alerts already? How is this different?

Architects: We do have alerts. Our DevOps have started strengthening our existing alerting and monitoring. We had missing alerts, incorrect configuration, etc. Sometimes alerts do not fire as the thresholds are kept low to minimize false alarms. API monitoring is not to replace existing monitoring systems, but it will rather complement and enhance it. A larger goal is also to go beyond monitoring to observability.

Me: What is Observability? Observability and monitoring are often used interchangeably.

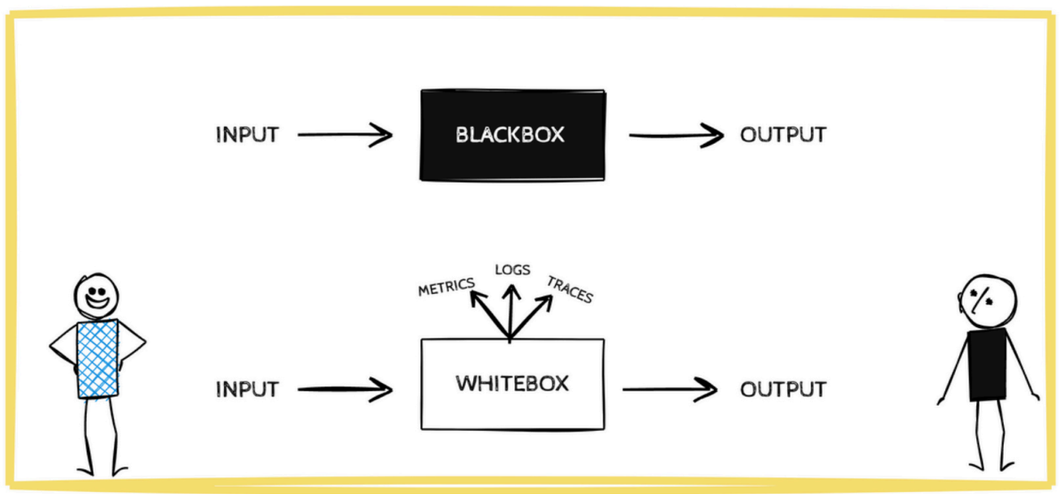
Architects: Observability is the ability to understand the internal state of a system by capturing and analyzing its outputs. Monitoring and observability go hand-in-hand. Monitoring enables observability. Monitoring is a subset of observability. While monitoring alerts us when something goes wrong, observability helps us understand why an error occurred and how to fix it. Observability provides granular, contextual insight into failures by using logs, metrics, and traces.

Me: API monitoring alone will not be able to enable observability correct?

Architects: Right, API monitoring alone will not suffice, it's just one aspect of observability. We will focus on the white-box part of monitoring and you and the testing team focus on the black-box side.

By combining our efforts, we will be able to achieve observability ! !

BLACK AND WHITE - THAT'S RIGHT!



Blackbox monitoring (a.k.a Synthetic Monitoring) focuses on end-to-end scenarios from the perspective of the user. It refers to monitoring the system from outside. Scripts are used to simulate the end-user's path. It gives insight into the behavior of the applications and services from an external standpoint.

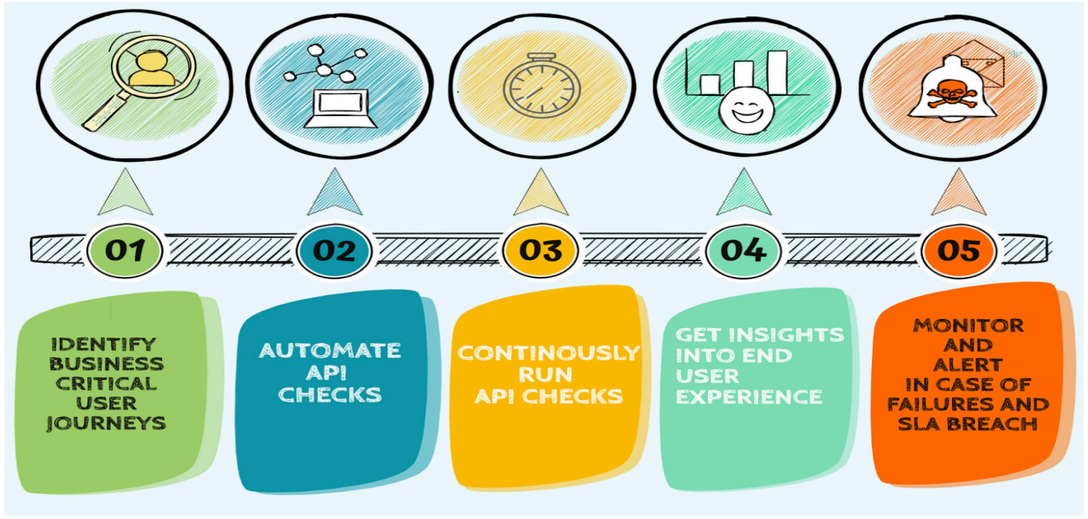
We cannot see through black; it is opaque. In a Black Box, you don't know what's happening inside the system. Blackbox monitoring can tell you when something fails. But cannot provide the information necessary to explain why it occurred. A black-box approach will not be able to recognize the root cause of the problem. That's where whitebox monitoring (a.k.a Real user monitoring) comes in.

Whitebox is obviously the opposite of Black Box. It's transparent and you can see through it. Using white box monitoring, you know the internals of the system with instrumentation (requires some modifications to your code to understand its inner state) to emit telemetry (metrics, logs, traces, etc.). In observability logs, traces, and metrics are collected across your entire IT infrastructure to assist teams in debugging when there is an incident.

Blackbox monitoring can help you understand "what" is broken from a user perspective. It focuses on the "what" of a problem, you need whitebox monitoring to find out the "why" of the problem. A white-box monitoring system will warn you if your database will run out of disk space in two hours. A black-box monitoring system will alert you if your API/service is down and can't be accessed by users.

While Whitebox monitoring can provide insight into your application, it cannot give you an accurate picture of what your users are experiencing. Blackbox monitoring proves valuable in this situation. To get the best results, you must incorporate both whitebox and blackbox monitoring. Most experts recommend combining heavy use of white-box monitoring with modest (but critical) use of black-box monitoring to achieve observability.

API MONITORING: THE STRATEGY



An API monitoring strategy must be tailored to fit the needs of the organization. API monitoring can include internal APIs, customer-facing APIs, and third-party APIs that are critical for the business. Here is how we achieved it at MoEngage.

STEP 1 > IDENTIFY BUSINESS-CRITICAL USER JOURNEYS

- User journeys refer to the series of interactions a user has with your product to accomplish a goal or satisfy a need. Identify the paths customers would take through your product that give the most business value.

- Determine which APIs are part of these journeys and prioritize them to automate and set up monitoring.

STEP 2 > DESIGN THE API CHECKS

- Start with the simple check if individual APIs remain healthy over time.

- The basic purpose of API monitoring is to ensure individual APIs continue to work as expected over time. You need to verify that all of your APIs return a 200 OK and that the response time is as expected. Perhaps a few more basic checks specific to each API.

- Check if existing API checks or a subset of them can be reused for monitoring.

- Monitoring individual API endpoints for response times and a simple 200 OK is a good start, but it does not suffice.

Chain up APIs to simulate the business-critical user journeys

- After you have some simple monitoring checks set up, automate the user journeys you have prioritized.

- To do that you need to run multiple. APIs in a particular sequence. This is called API chaining where the output of one API request serves as the input To do that you need to run multiple. APIs in a particular sequence. This is called API chaining where the output of one API request serves as the input

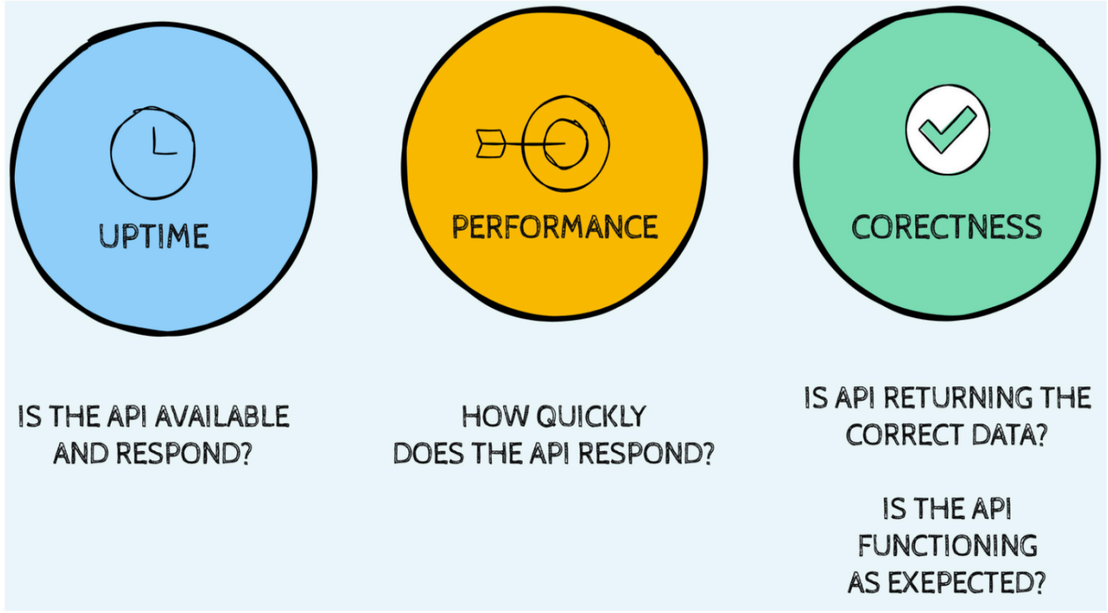
- Tests that provide quick feedback are suitable for API monitoring. To make long-running time-intensive checks work, they need to be split intelligently.

Uptime, performance, and Correctness

- Your API monitoring should be can to answer the following questions:

- APIs available and responding within the promised SLA?
- How are your APIs behaving?
- Are your APIs functioning as expected?

- The three fundamental things you need to consider are Uptime, Performance, and Correctness. Correctness here is not limited to the data validation of API response. Also, "correctness" refers to checking complex workflows.



STEP 3 > CONTINUOUSLY RUN API CHECKS

- Run automated tests continuously at short intervals Depending on the business context and API's criticality, API monitors can run every minute, every hour, or at any frequency you choose. At MoEngage, we run individual API health checks every few minutes, and complex API workflows every few hours. We use API monitoring to verify API functionality, responses, service level agreements (SLAs), etc.

Monitor non-production environments

- Although API monitoring is typically done after an application is deployed, it can also be performed during development or staging. Monitor non-production environments(Staging, QA, etc) as it allows you to catch API errors faster and prevent issues that could negatively affect the team's productivity.

Monitor from different geographical locations.

- To collect, share, and store data, many laws and regulations are in place. Data storage outside national boundaries is prohibited by many nations. To comply with this regulation, businesses do not have to store data in different data centers in different geographical locations. You should monitor your APIs from where your customers interact with your services and applications. This gives you a better idea of what your customers are experiencing.

API monitoring tools typically support a wide range of geographical locations enabling you to measure network latency from many worldwide locations.

Pick the right tool for the job.

API monitoring tools are abundant in the market. If you're a beginner, my recommendation is Postman. First of all, most testers are familiar with Postman, so there isn't a steep learning curve. Your collection can be turned into a monitor.

With Postman API monitoring, you can easily stay on top of the performance and health of your APIs and services. It's easy to set up API monitoring in Postman. If you do not like the limitations and costs of Postman monitoring, you can also use Newman, a command-line interface (CLI) version of Postman.

STEP 4 > GET INSIGHTS INTO END-USER EXPERIENCE

To get insights into the end-user experience, share API performance data with observability tools of your preference. For instance, Postman integrates with observability tools, such as Datadog and New Relic, so you can correlate data from your Postman Monitors with data from other telemetry tools. This is where the Blackbox and Whitebox monitoring tools communicate to enable observability and help you on-call to debug the issues on failures.

Observability and monitoring tools are designed to work together to give you insight into the health of your IT infrastructure. By integrating monitoring tools with observability tools, developers can diagnose and fix performance problems related to APIs proactively. Observability tools identify the underlying cause of an issue, while API monitoring tools trace it back to an API.

STEP 5 > ALERTING AND MONITORING

You won't be able to monitor your APIs if you have to check the status of each monitor manually. Set up automated alerts to notify engineers/ on-call when functionality is broken, APIs are down or slow, SLAs are breached, etc.

- You can decide how you want to be notified in case of a failure. PagerDuty(phone call), Email, Slack, SMS, etc. Webhooks can also be used to integrate your tool with tools it does not support.

- In case of an incident, the person from a team who is currently on-call receives the incident alert.

- API Performance data can be visualized using a dashboard: Postman displays the results of every monitor run on a dashboard. To troubleshoot more effectively, the dashboard can be scoped to individual requests, run types, and results.

- Grafana for visualization and alerting purposes, backed by data source Prometheus. Grafana allows you to combine Blackbox and Whitebox monitoring data in the same dashboard to get a comprehensive overview.

- You should refactor flaky tests immediately or completely replace them with a simpler test that doesn't raise false positives. Otherwise, API monitoring could produce false positives while omitting other potentially serious issues. To make API monitoring work, flaky checks must be eliminated vigilantly. Oncall shouldn't receive an alert in the middle of the night, but nothing is wrong with your services.

CONCLUSION

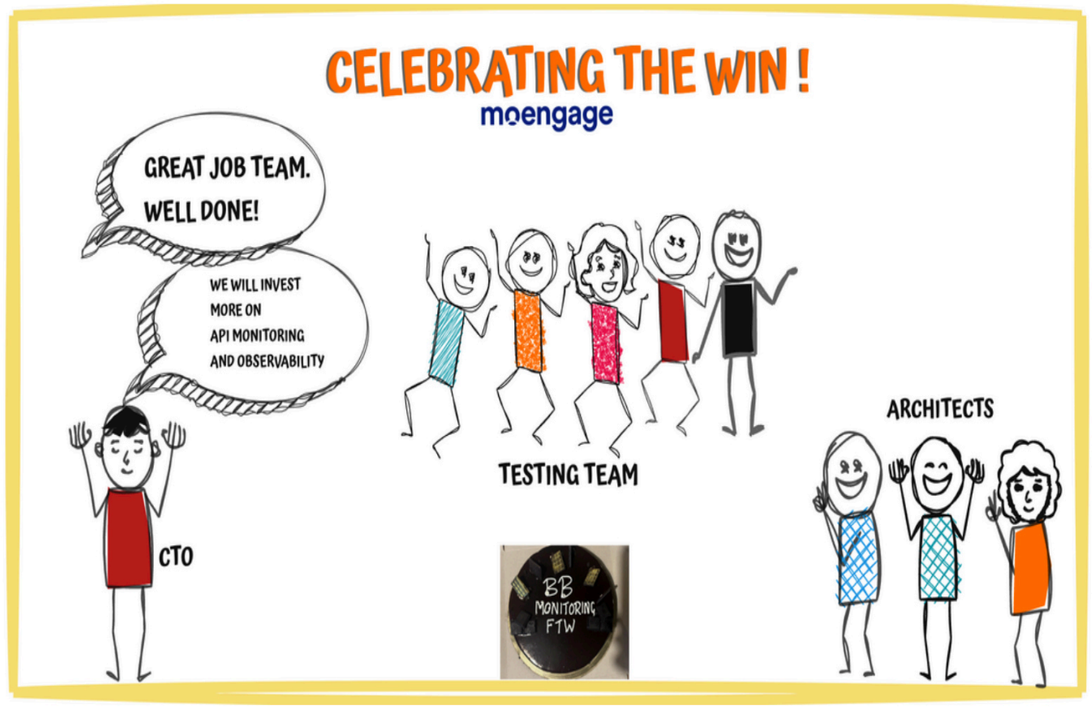
Can you identify issues before they hurt your customers?

Yes, you can with API monitoring!

Shift-right (testing on production) is just as important as switching left (early testing). API monitoring facilitates continuous testing in production.

Let's face it, no team is immune to bugs. API monitoring and observability can prevent bugs from impacting customers through proactive detection and troubleshooting. In addition to saving money and time, you can keep your customers happy. In the last six months, at MoEngage, we have discovered 25 production issues through API monitoring before customers could escalate them. In most cases, they were not code issues, but related to our IT infrastructure or scaling issues. At MoEngage, we like to celebrate small wins. Our CTO threw a party to congratulate the testing team and promised to invest more in this project.

Improve customer satisfaction, increase revenue, and ship code faster and more confidently. Invest in API monitoring and observability!



- PRASHANT HEGDE

Prashant is an empathetic leader who finds fulfillment in helping others succeed.

He enjoys sharing his experiences by blogging and contributing to software testing communities worldwide. Prashant heads the testing unit at MoEngage, a leading insights-led customer engagement platform. Prashant is an avid blogger and a frequent speaker at software testing conferences.

Connect with Prashant via -
LinkedIn: https://www.linkedin.com/in/prazhegde/
Blog: https://www.prashanthegde.biz/articles
Youtube: https://www.youtube.com/channel/UCdWXf_z7S9l4yjMAki_ERfw

QUALITY TALKS: AN EXPERT PANEL SERIES BY ACCENTURE QES DACH

QA TIME: CONVERSATIONS WITH QA LEADERS - A QASE VIDEO PODCAST

RECOMMENDED EVENTS AND YOUR CHANCE TO MEET TEAM TEA-TIME WITH TESTERS IN 2024

01

TEST AUTOMATION DAYS 2024
MAY 29-30, 2024

CHECK IT OUT!

02

SOFTECASIA 2024
SEPT 10-11, 2024

CHECK IT OUT!

03

RTC 2024
JUNE 12-14, 2024

CHECK IT OUT!

TEA-TIME WITH TESTERS

JOURNAL FOR NEXT GENERATION TESTERS

CONTENT PREVIEW : ISSUE 02/2024

MORE AWESOMENESS IS ON YOUR WAY THIS SEASON!

01

UNLOCKING POSSIBILITIES IN TEST AUTOMATION WITH CHATGPT

An insightful timely read by Geosley Andrades .

02

TEA AND TESTING WITH JERRY WEINBERG

We are bringing back the treasure of knowledge that Jerry Weinberg has left behind for us. More awesomeness on its way....

03

MICRO HABITS: A SYSTEM FOR ENGINEERING YOUR EFFICIENCY

A thought-provoking piece by Nitin SS.

INTERNATIONAL JOURNAL FOR NEXT GENERATION TESTERS

TEA-TIME WITH TESTERS

THE SOFTWARE TESTING AND QUALITY MAGAZINE

Created and Published by:

Tea-time with Testers.

Schloßstraße 41, Tremsbüttel

22967, Germany

Editorial and Advertising Enquiries:

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

Be with us and visit our website:

www.teatimewithtesters.com