No. 01
November 2024

testing experience

Reviving Testing Experience: *A New Era Begins*

The Importance of Accessibility in the Modern Digital World

A Learner's First Few Steps for Exploring LLMs

Listen beyond the pass and fail

The Road Less Taken

AGILE
TESTING DAYS

NOV. 24 - 27, 2025

SAVE THE DATE
SAVE THE DATE
SAVE THE DATE
SAVE THE DATE
SAVE THE DATE

REGISTER NOW

agiletestingdays.com

Reviving Testing Experience: A New Era Begins

Sometimes, life has a way of leading us back to unfinished business. For a long time, I've wanted to revive **Testing Experience**, but I needed that final push to make it happen. Now, that moment has arrived.

For those who remember Testing Experience from its heyday between 2008 and 2014, ***welcome back!*** And for our new readers, we're thrilled to introduce you to what we believe will become an essential resource in the testing community. Back in its prime, Testing Experience was a market leader, with over 30,000 readers per issue and some editions being downloaded more than 400,000 times over the years.

This first issue of our new era is packed with insightful articles from renowned speakers and respected professionals in the field. For those attending **AgileTD**, you'll even have the chance to meet some of these authors in person and discuss their articles face-to-face.

While the idea to revive the magazine was mine, as is often the case, the hard work was done by others. I've gained a bit of a reputation for that! I want to extend my heartfelt thanks to all the authors, partners, and advertisers who made this print edition possible. Your support is invaluable, and it's not something I take for granted. A special thank you goes out to my incredible team at AgileTD, especially Yalhen, Jana and Marc.

Testing Experience remains freely available, and our aim is to keep it that way as long as we can sustain it financially. In 2014, we had to stop publishing due to the high costs involved. This time around, we need your support to help this magazine reach the wide audience it deserves. Please, share it with your networks.

If you have a topic you'd like to see covered, feel free to reach out to us at editorial@testingexperience.media. You can visit the magazine's website at testingexperience.media

We hope this issue brings you joy and that you enjoy reading it as much as we enjoyed putting it together.

With warm regards,
José

Impressum

EDITOR

trendig technology services GmbH
Kleiststraße 35
10787 Berlin
Germany

Phone: +49 (0)30 74 76 28-0
Fax: +49 (0)30 74 76 28-99

E-mail: hi@trendig.com
Website: www.trendig.com

EDITORIAL
José Díaz

LAYOUT & DESIGN
Yalhen Rudolph

WEBSITE
www.testingexperience.media

ARTICLES AUTHORS
editorial@testingexperience.media

ADVERTISEMENTS
editorial@testingexperience.media

PICTURE CREDITS

CONTENT

Auto-Observability: The Power of Observation

Author: Alex Schladebeck

COME WITH ME TO AN ESCAPE ROOM

My best friend and I love doing escape rooms. We're also utter geeks, so we spend time talking about how we do escape rooms as well. We realised that our strength is in our ability to speak out loud about what we're seeing and thinking as we're looking around the room. "There's three pictures with monkeys here on the wall", "I'm seeing statues of zoo animals on the shelves here", Even as we're making sense of what is going on, we're already sharing our train of thought with each other.

CURIOUS VERBOSE OBSERVERS

I think my best friend would make a great tester. She shares qualities that I see in good testers – she is observant, she is curious about the meanings, workings and systems of the things she observes, and she is able to talk about her thought processes. Of course, she would probably be a good tester even if she weren't verbose – however, she would probably not be able to systematically identify patterns for herself, and she'd find it hard to teach others how to improve or practise.

NARRATING AS A WAY TO PIN DOWN INTUITION

It always bugged me that "intuition and experience" was the answer to improving at exploratory testing. It's not particularly actionable. There must be better approaches than just waiting patiently (patience is not one of my virtues). There were some approaches out there – James Bach describes *Testopsies*, as a way of improving by watching a testing session to observe the work going on. And I started working with Huib Schoots on encouraging people to actively narrate their work while they are testing. "What am I doing?" "What am I thinking" "What does this remind me of" "What assumptions do I have". Narrating makes us more aware of our thoughts and also lets others benefit from our stories and hear (and correct) our assumptions. I started doing it more and more, and really getting interested in how we explore.

MICROHEURISTICS

Working in this way led me to "discover" (or describe) *microheuristics*. My big aha moment was when I realised that quite often, two testers pairing will have the same idea for what the next step or action could be while they are performing exploratory testing. This happens even if that next step is not a "logical" next move (e.g. saving after completing a dialog). This was intriguing. If they're not following a script, how do two testers (who – in the cases I observed – neither work in the same company nor on the same domain) come to the same idea at the same time?
My suspicion was that we have shared models, assumptions and

Alex Schladebeck
CEO and Quality Advocate
@ BREDEX GmbH

patterns that result from learning about testing, knowing about software, experiencing errors, and hours of testing (either hands-on, or automated, or simply asking questions). These things come quickly to mind when we see something that reminds us of them – and give us the idea of what to do next (probably to prompt interesting behaviour). My hope was that by learning to talk about these and by practicing narration during testing, we would have some concrete patterns that we can describe and use to teach others.
My name for these patterns is *microheuristics*.

The definition is:
A quick way to determine "what's my next action" while testing. A microheuristic is our brain applying what we've just learned to decide on the next step or experiment. The result of a microheuristic being applied will usually prompt an immediate action. Such actions are rooted in snap judgements that we make explicit and strategic by describing them.
They are heuristics because they help us (in a fallible way) to make a decision. I called them "micro" in comparison to the well-known testing heuristics which more often guide us to decisions about charters or risks – not to the very next action in a testing session.

"In order to improve myself and to train others in the role, I needed to get better at talking about my patterns."

One example is the "pimple" heuristic:

> *If something seems problematic / acts oddly, interact with it directly and indirectly to evaluate it more closely, in different situations.*
> *The interaction can be editing, searching for or redoing steps.*
> *Keep poking until something comes out, or you're satisfied it won't.*

The whole story and the other microheuristics I've described can be found here: schladebeck.de/microheuristics/

NEW ROLE, WHO DIS?

I felt pretty pleased with the concept of microheuristics. It was a good way of describing my experience to others who share my passion for testing. And then I became a manager, which presented a new challenge - I didn't have heuristics for this. And similarly to looking at an experienced tester doing their magic and not understanding it, I was faced with figuring out how to lead in all of its joyous complexity using the black box of those around me as a guide. Not all managers are good at explaining their thoughts either.
After a few years, I did start to get intuition and experience, only then to realise that I was at the same place again. In order to improve myself and to train others in the role, I needed to get better at talking about my patterns. Ideally I wanted something as concrete as microheuristics, but I haven't found that (yet?). Instead, I went back to the principle that led me to them. Narrating. Being explicit about those firing neurons in my brain. And I discovered that what I'd been doing all this time was a kind of auto-observability. I've been using narration as a way of following events through my brain, of adding log statements that I can find later. In this sense, I'm the system that I'm asking new questions of. I'm looking at my output to determine my inner workings. It's a bit meta but it feels like a good fit.

Since I enjoy having names and labels for things, finding the term "auto-observability" was a huge relief. It didn't come from my own brain cells though - I attribute a great deal of it to Elizabeth Zagroba who suggested the term "leadership observability" and sent me to Hazel Weakly's blog on "redefining observability"[1]. Hazel ends with the point that observability is organizational learning, and notes that "observability is the process through which one develops the ability to ask meaningful questions, get useful answers, and act effectively on what you learn". Of course this is relevant for the observability we talk about in a technical sense. But it's also true of leadership work. We become the system that others ask questions of.

CAN WE *OBSERVE* OUR LEADERSHIP ACTIVITIES?

I think that using this approach can help us to *observe* (in both senses) our leadership abilities. I started gathering and labelling activities on a whiteboard of *what I am doing* when I do leadership and management work. Activities *like making decisions, deciding whether to even make a decision, prioritising for myself, prioritising for the company, managing conflict, dealing with unknown circumstances, dealing with crisis, staying sane (that's an important one!).*

I think this idea of auto-observability is a powerful one that can be taken into any role.

And for each one, I'm collecting principles and experiences and heuristics of how to go about them. They're not microheuristics, because seeing a thing doesn't determine a next action. But they provide ways to talk to others about how I'm working. Which gives the same benefits as narrating testing: I learn, they learn, and we have the opportunity to find holes in our assumptions.

APPLICATIONS

I think this idea of auto-observability is a powerful one that can be taken into any role. If you're new at it – ask others around you how they do their work. Start categorising what you're doing and collect your thoughts and experiences on those activities. And when you need to mentor, train and support others – practice talking about your thought processes. Be the person who explains what is going on in the black box, be it for complex debugging, locating performance problems, prioritising risks or managing communication and conflict. Whatever it is you're doing, try to know why. I think testers are probably good at that. After all, we are the curious observers. We just need to get verbose.

I'll leave you with a quote from a great author that this topic always makes me think of:

> First Thoughts are the everyday thoughts. Everyone has those. Second Thoughts are the thoughts you think about the way you think. People who enjoy thinking have those. Third Thoughts are thoughts that watch the world and think all by themselves. They're rare, and often troublesome. Listening to them is part of witchcraft.
>
> – **TERRY PRATCHETT, A HAT FULL OF SKY**

1. https://hazelweakly.me/blog/redefining-observability/

Listen Beyond the Pass and Fail

Author: Lena Pejgan Nyström

Lena Nyström
Engineering Manager
@ Nordnet

Sometimes, it's hard to see the forest for all of the trees. We can't see progress, or stagnation, if we only ever look at the current state.
Years back, I took on a new role with the main goal being to move the organisation's testing to the next level. In particular - their automation strategy. It was sold to me as a solid state where the department had invested heavily in automation, had a lot of automated tests in place and had worked really hard on making automation a part of the normal team delivery. I ́ve worked with my fair share of problematic automation transformation projects and I like to believe I know a bit about what not to do.
I envisioned a field of green tests and a mindset of "If it fails we fix it" and teams, and testers, working together to constantly improve. That is not really what I saw.
I saw failing tests, an increasing number of tests, testing still being a bottleneck and worse: it felt like no one was worried. I did. But I couldn't see the pattern by only looking at individual data, like a single conversation or a single nightly run. So I started digging. I pulled historic data into spreadsheets, I started asking questions and slowly pieced together a picture of what was going on, and what we could do about it.

I talked to people who had come the farthest and worked my way outwards. I spoke to testers, scrum masters, developers, architects, product owners, operations and management. The questions shifted but they all circled around if people/teams felt confidence in the automation, how much time they spent on maintaining them vs. improving them, what their biggest wins and pain points were.
Some things stood out.

- Testers were too busy delivering to make space for proactive work.
- Testers felt the main problems were impossible to fix, better to accept and work around them
- You get what you measure, and you improve what you are being measured on.

TOO BUSY DELIVERING TO MAKE SPACE FOR PROACTIVE WORK.

If you don't actively make space for it - bigger refactoring and improvement work can be very hard to fit into a busy schedule. There will never be time for it naturally. Parkinson's law states "work expands so as to fill the time available for its completion" - and this is what I saw here as well. People were too busy to. What I saw here was a never-ending cycle of

- Analysing the last run
- Debugging issues to decide if they were bugs or problems with test data or test environments
- Fixing issues, or more commonly - mark the test as having a known bug
- Setting up for nightly run

Repeat everything the next day. Any time left was used to write new automation. Little to no time was spent on improving things (proactive) only fixing things (reactive). As a result, completing this cycle took more and more time, leaving less time for improvement and requiring more man- and computer power.
When you are in this loop – it is really hard to see it. It is very similar to how we react to stress. The more stressed we are, the less access we have to our full capabilities[2].

PROBLEMS ARE IMPOSSIBLE TO FIX, BETTER TO WORK AROUND THEM

When you don't know about an area, it can feel impossible to change anything about it. My testers kept saying the issues were "impossible to fix" and instead tests were re-run locally - where they often passed. So they weren't prioritised. When looking a bit deeper, there were a few areas where we could fix things, with a little help from others.

When you don't know about an area, it can feel impossible to change anything about it.

We had problems related to timing (backups, patches, batch jobs) or hardware/network (firewalls, DNSs, queues). They *felt* unfixable, but with collaboration with our operations people we worked on rescheduling tests, re-configuring hardware and buying more computer power.
Problems with data limitations, limited capacity of our environments and dependencies to other services could be improved by changing how we worked. We moved to generating data as needed instead of leaning on pre-existing test data, we worked on removing dependencies and put in place a long term plan for getting the resources and prerequisites needed for a good, modern setup of environments and data. Some of our problems were due to very old setups and hardware but the general group seems to still be relevant, based on the conversations I'm still having.

Some things were simply problems related to communication and/or process. A lack of communication between teams or teams having different timelines and priorities. Team A didn't make time to fix their tests to match changes made in Team B's code because they were busy doing something else. Team B had no idea

2. https://dandypeople.com/blog/stress-in-a-nutshell-and-the-connection-to-leadership/

their changes would affect Team A so they never thought to inform them. Or they assumed someone else would. The hardest problems to solve are people problems, but the solution typically starts with talking to each other.

YOU GET WHAT YOU MEASURE. YOU IMPROVE WHAT YOU ARE MEASURED ON

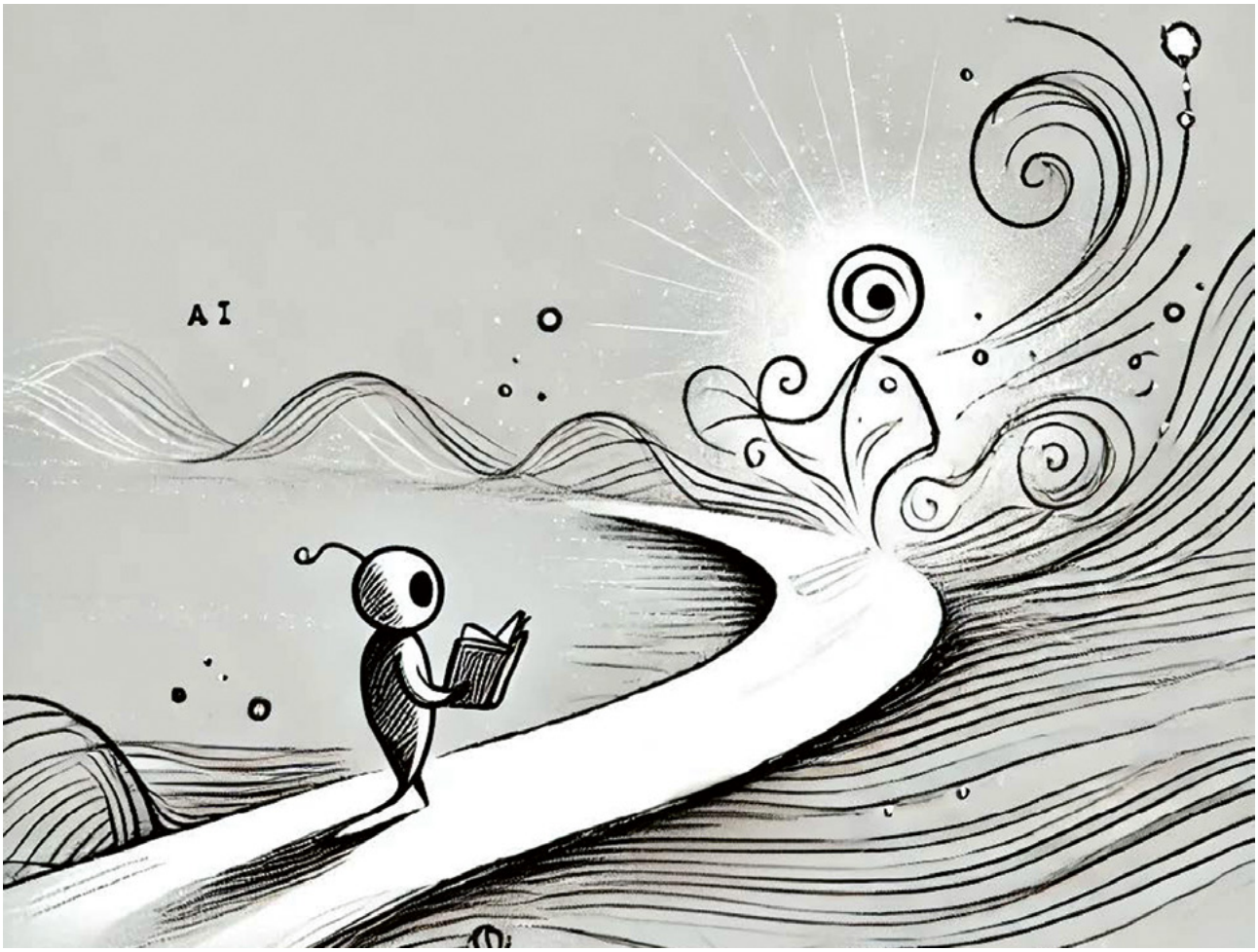
I found an unexpected resistance to deleting tests. We had tests that had run for years without ever passing. People still did not want to delete them.

"Make sure you make room for continuous improvements. Making time for proactive work will save time in the end."

One reason is the emotional barrier to removing something you have invested in - sunk cost fallacy. "It might come handy later". This was not surprising, although I was underestimating the strength of it.
More surprising was the realisation that one of the main metrics we measured, and reported, was the number of test cases automated. It's an easy thing to measure, but a hard thing to draw any conclusion from. It says that we are prioritising automating tests but says nothing of the value added or the quality of the automation. On a department level this might make sense, particularly in combination with things like total run time, time spent on maintenance, speed of delivery and my favourite: confidence in the automation. But on an individual level it had very bad effects. When you, a tester, are setting yearly goals and automating tests is a prioritised activity specifically mentioned in the career framework of your company - would you delete tests? Or would you make sure to prioritise creating more, to meet those goals? And as a manager or lead - If you have to report those numbers on a monthly basis to a board - who expects to see big nice increasing bars in the graph - would you prioritise setting aside time to evaluate and delete tests? Or would you leave them as is, arguing that they might come in handy later.

SUMMARY

Looking at trends rather than just the current state will help you see patterns. Those patterns can help you identify areas of improvement. If you never take time to zoom out and look for those patterns, you risk getting stuck in reactive mode - only seeing the next work task to pick up. The degradation of quality, speed of delivery and value might be so slow it's hard to see it's there. It's like boiling frogs.
Involving people with different skill sets might solve problems you thought were unsurpassable. Impossible might be possible with another set of tools. Think about why it feels impossible. Who would you need to be to be able to solve it? What would have to change for it to no longer be impossible?
Make sure you make room for continuous improvements. *Making time* for proactive work will save time in the end. The money and time saved and the increased team and customer satisfaction is a solid business case any day. It might be surprising, but no one (not even your manager) can create that time for you. Time is a limited resource, work is unlimited. But you can get help making space for it - but only if people know you need it.
And lastly: You get what you measure, and people will improve what they are being measured on. So be careful what you go looking for.



Picture created with AI

A Learner's First Few Steps for Exploring LLMs

Rahul Verma
Head of Test Automation
@ trendig technology services GmbH

Author: Rahul Verma
Co-authored with ChatGPT o1-preview.

Alright, so you've decided to dive into the world of Large Language Models (LLMs), have you? Bold soul. I've been navigating this maze myself, so I figured I'd share my journey. It's been going well for me - when the stars align. But don't let me cramp your style. Go ahead, blaze your own trail. After all, who am I to steer your ship? To each their own, as they say. Which is polite for 'I wouldn't do that but go ahead, poke the bear. I'm sure that it's in a good mood today.'

1 START WITH A FRONTIER MODEL'S WEB INTERFACE

First things first, dip your toes in. Head over to a frontier model's web interface like ChatGPT or Claude. Start small. Ask it to craft an ode to your morning alarm clock or explain the secrets of the universe as told by a toddler. You know, the usual existential quests. It's all fun and games until the machine starts pondering your existence. Just kidding. Unless it already has. Probably.

Your goal at this stage is to grow beyond gotcha prompts that one borrows from the university of LinkedIn like “Which is larger - 9.11 or 9.9?” or “How many r’s are there in Strawberry?”. Wrap that up. Fast. Have one more go if you must. Have one last laugh. Then start asking questions that actually matter to you. Like requesting a love letter to your cat or a heartfelt apology to your neglected gym membership. At least that's personal and beats indulging in pseudo-intellectual games. It's far more rewarding than trying to stump the machine with overused riddles that only serve to inflate egos on LinkedIn.

2 PAY FOR IT AND GO DEEPER

Once you've had your fun and realize the abyss of curiosity is deeper than you thought, consider parting with some cash. Yes, open that wallet. An investment in knowledge pays the best interest - or at least gives you something to talk about at parties. Dive deeper. Learn about prompt engineering - a term that makes "asking nicely" sound like a science.

Try different types of problems and play around with input-output formats. Maybe even learn about Transformers and the Attention mechanism. No, not the robots or your tendency to zone out when someone mentions 'synergy.'

3 EXPERIMENT WITH ROOT PROMPTING

Time to get a bit more sophisticated. Experiment with root prompting - the first prompt that sets the tone for the entire chat. Think of it as your opening line at a party. You wouldn't walk in and shout something inappropriate - well, maybe you would, but let's aim higher. It's like your first tweet of the day: make it count before you're cancelled for that typo that changed "public" to something else entirely.

Maybe go with, "You're a motivational speaker who uses sarcasm to inspire," and see how the AI motivates you. Or perhaps, "You're a journalist who can't resist a good pun." The idea is to set the stage so the AI knows which costume to wear. Or don't bother. It's your rodeo; I'm just the guy selling popcorn.

4 CREATE A CUSTOM GPT

Why keep hammering out the same old commands when you can get the AI to read your mind - or at least pretend to? Noticing that your prompts are starting to feel like a scene from Groundhog Day? Saying the same thing over and over, like a sitcom rerun no one asked for? Repeating yourself like a parrot with short-term memory issues? It's time to build a Custom-GPT.

Turn those repetitive requests into custom instructions. It's like teaching a parrot to talk, but without the danger of it quoting your questionable late-night texts. Let the AI handle the repetitive stuff so you can ponder life's mysteries, like why socks always go missing in the wash.

5 EXTEND CUSTOM GPT WITH CODE AND TOOLS

Feeling experimental? Upgrade your Custom GPT with some code or outside services. Add a few gadgets into the equation. How about connecting it to your smart mirror? Who knows? It might start giving you motivational speeches - or brutally honest critiques - every morning. The possibilities are endless - until you develop a complex.
By integrating code and external services with your Custom GPT, you're essentially giving it the keys to your digital kingdom. It's like handing over your house keys to a stand-up comedian - you don't know whether you'll come home to a surprise party or find your furniture rearranged for a joke.
Remember - with great power comes great potential for hilarious mishaps.

6 EXPLORE THE API LAYER

Now we're stepping into the big leagues. Time to dive headfirst into the API layer. Don't panic; it's not as scary as it sounds. Start tinkering with the Chat API - send messages, receive responses, maybe even upload some images. Yes, even that unflattering selfie you've been hiding since the last family reunion. Who knows? The AI might appreciate your unique sense of style. Play around with response formats— make it sing, dance, or at least reply in JSON. Enable tool or function calling; make the AI fetch data,

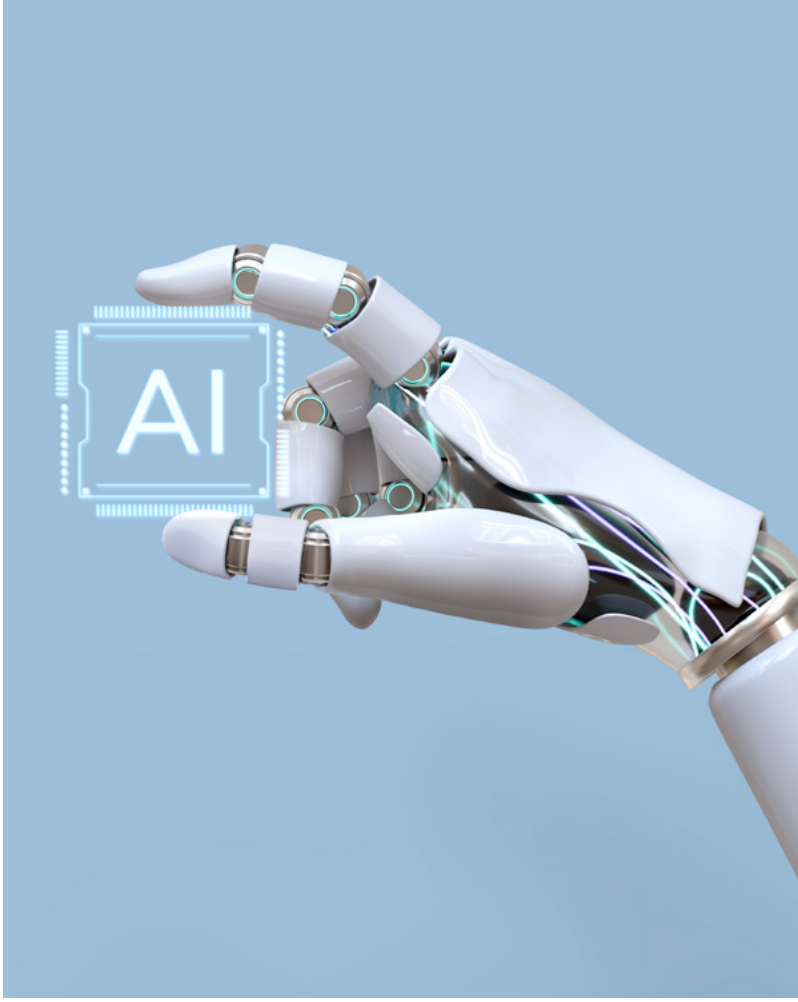
perform calculations, or tell you the weather in Timbuktu. Manage context like you're conducting a conversational orchestra, keeping every instrument in sync.

But let's not get ahead of ourselves. This is still basic territory. So, maybe keep that cape in the closet a bit longer. You're not quite ready to audition for "America's Next Top Coder." Master Yoda you are not - yet.

7 CREATE A MINI-FRAMEWORK

Fed up with the same old, same old? Time to whip up your own mini-framework. Craft some reusable modules to tackle the tedious bits, so you can focus on more crucial matters - like figuring out why you've got 200 unread emails or debating whether pineapple belongs on pizza.

And hey, when you've automated the dull stuff, you can finally catch up on that series everyone's been spoiling for you. Plus, automating repetitive tasks makes you look like a genius, even if you're just avoiding actual work. It's a win-win.



Designed by Freepik

8 EXPLORE LANGCHAIN AND LANGSMITH

By now, you might as well dive into LangChain and LangSmith. They're like the secret sauce for working with LLMs - or perhaps just more rabbit holes to tumble down. Either way, worth a peek.

But fair warning: LangChain can take a simple task and wrap it in layers of complexity so thick you'll need a machete to cut through. It's like asking for a glass of water and being handed a blueprint for a desalination plant. Sure, it's impressive, but all you wanted was a drink. On the bright side, wrestling with it gives you something to complain about on social media - because who doesn't love a good rant?

9 CHUNKING LARGE CONTENT AND SEMANTIC SEARCH

Why try to leap over a mountain in one bound when you can stroll up it one step at a time - and save yourself from cardiac arrest? Dealing with massive amounts of text? Time to learn about chunking and embeddings. No, it's not a new fitness regime or something you'd find in a dodgy nightclub.

Use Vector Databases for semantic search. Sounds fancy, doesn't it? But it's not as high-tech as it sounds. Break down that colossal content into bite-sized pieces. Summarize effectively. Then casually drop "vector embeddings" into conversation at parties. Your friends will be so impressed they'll probably change the subject.

10 DIVE INTO RETRIEVAL-AUGMENTED GENERATION (RAG)

At the end of this enlightening journey, it's time to tackle RAG - Retrieval-Augmented Generation. Start small; there's no need to wrestle a bear on your first day at the zoo; after all, even Picasso started with finger painting.

Try out different RAG styles, like a chef experimenting with recipes - some will be culinary masterpieces, others will set off the smoke alarm. Combine your accumulated knowledge to craft something truly magnificent - or at least something that doesn't make you want to throw your computer out the window.

BE SCEPTICAL AND THINK CRITICALLY

But hold on a minute. Before you get too carried away, remember to be skeptical. Wear your critical thinking hat - if you can find it under all that enthusiasm. There's no true learning or exploration without questioning what's in front of you. Especially with LLMs. They might dazzle you with eloquence, but they're just algorithms predicting text - not philosophers decoding the universe.
The LLMs can generate impressive text, but they don't understand context like a human does. They don't feel, they don't think, they don't ponder the meaning of life while sipping a cup of tea. So, while you're experimenting and building with LLMs, keep your wits about you. Question the outputs. Cross-check information. Just because it sounds convincing doesn't mean it's correct.

Think of LLMs as your chatty friend who always has an answer, even when they have no idea what they're talking about. Entertaining? Yes. Reliable? Not always. So, take everything with a grain of salt - maybe the whole shaker.
Wisdom isn't just about having answers; it's about knowing which questions to ask and which answers to trust.

CONCLUSION

So there you have it - a path for exploring LLMs that's been working for me. This should keep you occupied for a couple of months, or at least until the next big distraction comes along. Maybe it'll suit you, or maybe you'll forge your own path and make me look like I'm stuck in the Stone Age. Either way, it's all part of the adventure. Go on, place your bets. After all, what's the worst that could happen? Actually, on second thought, let's not go there.

Beyond the Mission: The Art of Adaptive Leadership

Author: Dr. Rochelle Carr

Leadership in today's dynamic business landscape goes far beyond simply interpreting a company's mission. It's an intricate art that fosters transformation and drives innovation - like being a juggler, but with more spreadsheets and fewer clowns. As change accelerates, effective leadership becomes an essential investment in both personal and organizational growth. What worked yesterday might not suffice today, making it crucial for leaders to navigate this landscape with a sense of humor and adaptability.

Designed by Freepik

To begin this journey, let's address the elephant in the room: fear. We all know it well, often leaving us pondering, "What do I really want?" Fear can cloud our vision, making it difficult to articulate our goals. Yet here's the catch: you can't achieve what you haven't named. So, grab a notepad (or a coffee-stained napkin - no judgment here) and jot down your aspirations. Think of this exercise as your personal GPS - minus the annoying voice telling you to make a U-turn. Naming your goals is the first step toward transforming them into reality.

However, it's easy to fall into the trap of unrealistic optimism. It's tempting to believe everything will fall into place like a scene from a rom-com, but spoiler alert: it usually doesn't. Leaders often fixate on deliverables, finding it easier to chase metrics than to nurture the people driving those results. Yet, it's essential to remember that people are wonderfully messy, coming with emotions, ideas, and the occasional existential crisis. Embracing this messiness is key; by focusing on your team, you cultivate resilience and innovation - like herding cattle, but occasionally discovering a calf genius who can solve complex problems.

When you realize you're stuck, don't panic. Stagnation can be a valuable teacher. Acknowledging that you're in a rut isn't a sign of weakness; it's a necessary step toward growth. This is where action steps come into play. Setting regular check-ins - whether through one-on-one meetings or team huddles - can create a space for open dialogue. These conversations should focus not just on deliverables but on personal and professional aspirations, helping to identify what truly motivates your team.

As you confront these barriers, remember to celebrate small wins along the way. These victories act as breadcrumbs guiding you out of the wilderness of stagnation. Whether it's resolving a long-standing conflict or completing a minor project, acknowledging these moments can reignite motivation and build momentum. After all, if we can't celebrate the little things - like finally getting the office Wi-Fi to work - how can we tackle the bigger challenges ahead?

Reflecting on the past can also provide valuable insights into shaping the future. What patterns emerge in your successes and failures? Analyzing these trends isn't merely about avoiding pitfalls;

it's about leveraging your strengths. Embrace what works, but don't hesitate to pivot when necessary - much like that diet you started last January. It's time to retire ineffective strategies and explore new avenues for success, which brings us to another important action step: embracing experimentation. Foster a culture where team members feel safe to share their "crazy" concepts without fear of immediate judgment - because sometimes the wildest ideas lead to the best innovations.

A powerful example of adaptive leadership in action is Netflix. According to Mark Fairlie, Senior Analyst & Expert on Business Ownership, Netflix originally started as a DVD rental service but quickly recognized the changing landscape of consumer behavior and embraced the shift to streaming. Instead of resisting change, they pivoted their entire business model. This transition required not only a strategic overhaul but also a focus on nurturing their employees. Their strong company culture prioritizes freedom and responsibility, allowing teams to innovate without the constraints of micromanagement.

Photo by freestocks on Unsplash

Netflix also exemplifies the importance of a feedback loop. The company encourages open communication, holding regular meetings to discuss what's working and what isn't. As they ventured into original content, they celebrated small wins, such as the success of "House of Cards," which validated their strategic shift. Importantly, they learned from failures too, like their misguided attempt to split DVD rentals from streaming services, using those lessons to refine their approach.

Another critical aspect of adaptive leadership is the power of diverse perspectives. Bringing in individuals who think differently isn't just trendy; it's essential for breaking through barriers. Surround yourself with people from various backgrounds, departments, or industries. These fresh voices can ignite innovation and challenge your thinking. Embrace the discomfort that comes with differing opinions - growth often lies outside our comfort zones. Plus, who doesn't want to argue about the best pizza toppings with someone from marketing? To facilitate this exchange of ideas, establish a feedback loop. This could be as simple as a suggestion box (digital or physical) or regular brainstorming sessions to gather diverse perspectives and insights about what's working and what isn't.

Finally, when faced with persistent obstacles, consider forgetting the barrier altogether. Sometimes, the best way to move forward is to chart a completely new course. This doesn't mean abandoning your goals; it means recognizing that rigidity can stifle creativity. Flexibility is a leader's best friend. If the road is blocked, maybe it's time to take the scenic route or simply grab a coffee and rethink the whole thing.

CONCLUSION

In conclusion, adaptive leadership in today's ever-changing landscape demands mindfulness, flexibility, and a good dose of humor. By naming your aspirations, engaging your team through regular check-ins, celebrating small wins, learning from the past while embracing experimentation, and creating a robust feedback loop, you'll foster an environment ripe for innovation and growth. So, roll up your sleeves, prepare to embrace change, and remember: the most effective leaders aren't those with all the answers, but those who bravely ask the right questions - and manage to have a little fun along the way. After all, if you can't enjoy the journey, what's the point of the destination?

Dr. Rochelle Carr

Path Forward Empowerment
Certified Coach, Speaker, Trainer
and Facilitator

A Test strategy for the Whole Team

Authors: Lisa Crispin & Janet Gregory

Modern software teams see the value of engaging the whole team to build quality in. We hear from a lot of people who want to get everyone on the team engaged in testing activities. They are often struggling with how they can plan and execute an effective testing strategy together. This led us to design the Holistic Testing model that can be used to design a test strategy for your team *(see image on the right)*.

Larger development organizations may decide to create testing strategies at different levels. This could include:

- A defined ubiquitous language for test definitions.
- A list of quality attributes that might be included in testing, with a definition of each.
- Guidelines for tools that are appropriate for different types of testing including test automation.
- Development approach used, for example, Kanban, Scrum, or one specific to the company.
- Product-level test strategy, including risks, constraints, and quality attributes to consider.

When a team is developing a new feature, they need to consider the higher-level strategies and build those constraints into their testing strategy.

Holistic Testing Model

The Holistic Testing model is particularly relevant for teams who are practicing continuous delivery/deployment or want to move towards that. It visualizes the major stages of the infinite loop of software development. In the image below, we've included examples of testing activities that may happen in the various stages. A strategy could be as simple as substituting the examples shown in the model below for your own testing activities at each stage.

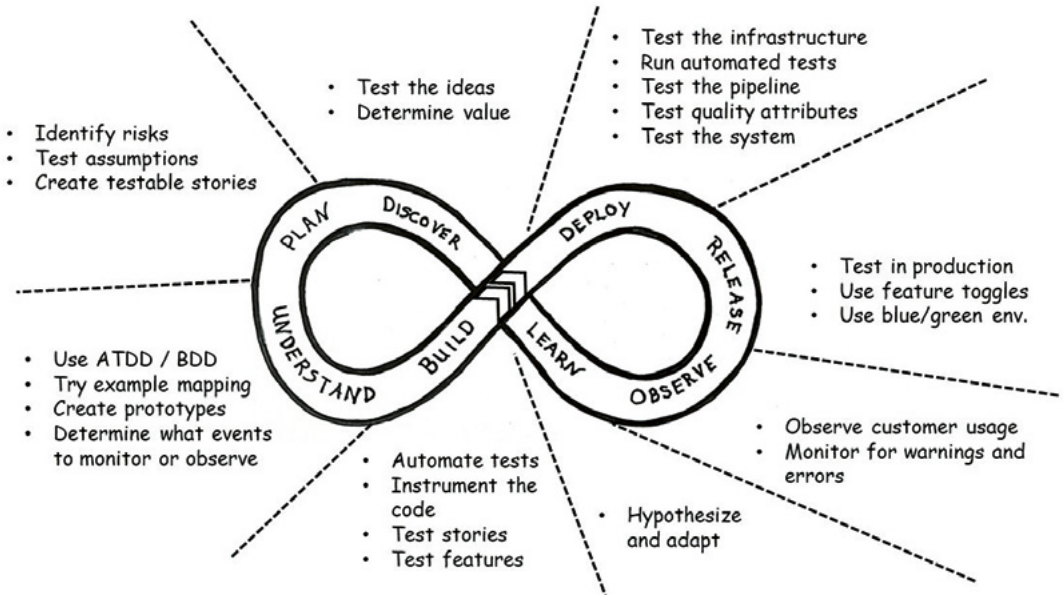
A new feature usually starts with discovery activities on the left side of the model. Step with us through the model to explore strategic choices at each stage, starting with the discovery activities.

DISCOVER

As business stakeholders think about what they want next, they need to be asking questions about what problems they are trying to solve, who the customers are that might be relevant, and what outcome they would expect. They need to test their idea and determine the value of the possible feature. Both Lisa and Janet have worked on delivery teams whose members were included in these early feature discussions. It's a great time to ask questions about the purpose of a new feature as well as learning how the business will measure the success of the new feature in production.

PLAN

Once the due diligence has been done, the delivery team can learn more about the feature(s) and what they need to build. They can plan by brainstorming, together with domain experts, what the feature might entail – this could be something simple like creating a feature mind map. The team could use a risk-storming session to determine what risks there might be, and perhaps produce possible risk mitigation strategies. Working with the business folks, the team can learn which quality attributes are most important – such as security or accessibility.



Part of planning is getting to the next level of detail – breaking the new feature into small, incremental stories. As a team, you could do story mapping, or maybe draw a simple flow diagram of the feature. Find visualization techniques that give your team and the product people the opportunity to understand how it would flow. This lets you start thinking about the core stories, identify complexities, and plan how to test them.

UNDERSTAND

As your team's development process proceeds around the loop, you may encounter a roadblock and go back to an earlier stage to regroup. It's better to get those roadblocks eliminated before moving on.

Once a team has sliced the new feature into stories, it's time to gain a deeper understanding of each one. There are many tools that can help drive out misconceptions and misunderstandings. Some of our favorites are Acceptance-Driven Development (ATDD) or Behavior-Driven Development (BDD), along with Example Mapping.(The book Discovery: Explore behaviour using examples by Seb Rose and Gáspár Nagy is one great place to learn about these). All of these are based on concrete examples which help show what assumptions people make. The examples can be turned into tests to guide development. They can be automated, as appropriate, and become part of the team's continuous integration pipeline, to guard against future regressions.

This is also a good time for teams to discuss what events, or monitoring information they need to build into the system for their observability and monitoring practices.

As part of the strategy, you would decide which of these techniques would be used. Of course, you may change as your team learns more about a particular story.

BUILD

By this stage, the team has most of the information about the stories that you are building, but not all. As a team works on a story, they learn more and adapt. Information that you gathered earlier is applied, such as instrumenting the for observability, monitoring, and/or analytics.

In our own experience, part of the test strategy has included pairing with the developer to review the tests at different levels, starting with the unit level. This builds another level of understanding. We have found that turning those examples from the earlier stages into executable tests helps the developers as they are writing the code. Fast feedback is so important. Pairing or ensemble (aka teaming or mob) programming are powerful techniques that could be part of your strategy.

As we move to the right side of the loop, the emphasis changes from individual stories and functionality to the system as a whole.

DEPLOY

The code has been deployed to a development or test environment. Now, the team can consider the types of testing that could not be done before. It's often the first opportunity to test quality attributes such as performance, load, security or accessibility. It's also time to think about the development pipeline. Infrastructure as code also needs to be tested. Do the automated tests run as expected? Do deployments to different environments succeed? The system as a whole – does it operate as expected? Does the right data get persisted?

RELEASE

Testing does not finish when your feature is released to the customer. The team should decide on release strategies, such as feature toggles or canary releases, which allow them to deploy to production while controlling whether customers can see the new changes. Part of your testing strategy is how to test the release. Can you safely test the feature in production? Or mitigate risk by using a strategy such as canary releases? The telemetry built into the new and updated code allows the team to keep track of production usage and identify anomalies quickly.

OBSERVE

How is your team going to monitor for warnings and failures? Can you do much of this in your early testing? If so, use the monitoring capabilities. All the events that are captured by the code's telemetry should be watched continually for indications that something might be wrong, or that your customers are using the system in unexpected ways. They can only be put in place when you think about them early – make them part of your strategy. Set up alerts with correct thresholds so the team can respond to problems immediately.

LEARN

With today's monitoring, observability and analysis tools, teams can gather so much useful information about how their applications behave in production. It's possible to see exactly how customers interact with the product. Teams can use this information to guide their next changes. Set a goal for the next step. Design experiments with hypotheses that include ways to measure the experiment's success. Even failed experiments help teams learn.

"Teams can use this information to guide their next changes. Set a goal for the next step. Design experiments with hypotheses that include ways to measure the experiment's success."

Documenting your testing strategy

Make your testing strategy visible so that you can discuss it not only within the team, but with stakeholders and other people involved in delivering new changes. It's important to use a format that is easy to update, since your strategy will continually evolve.

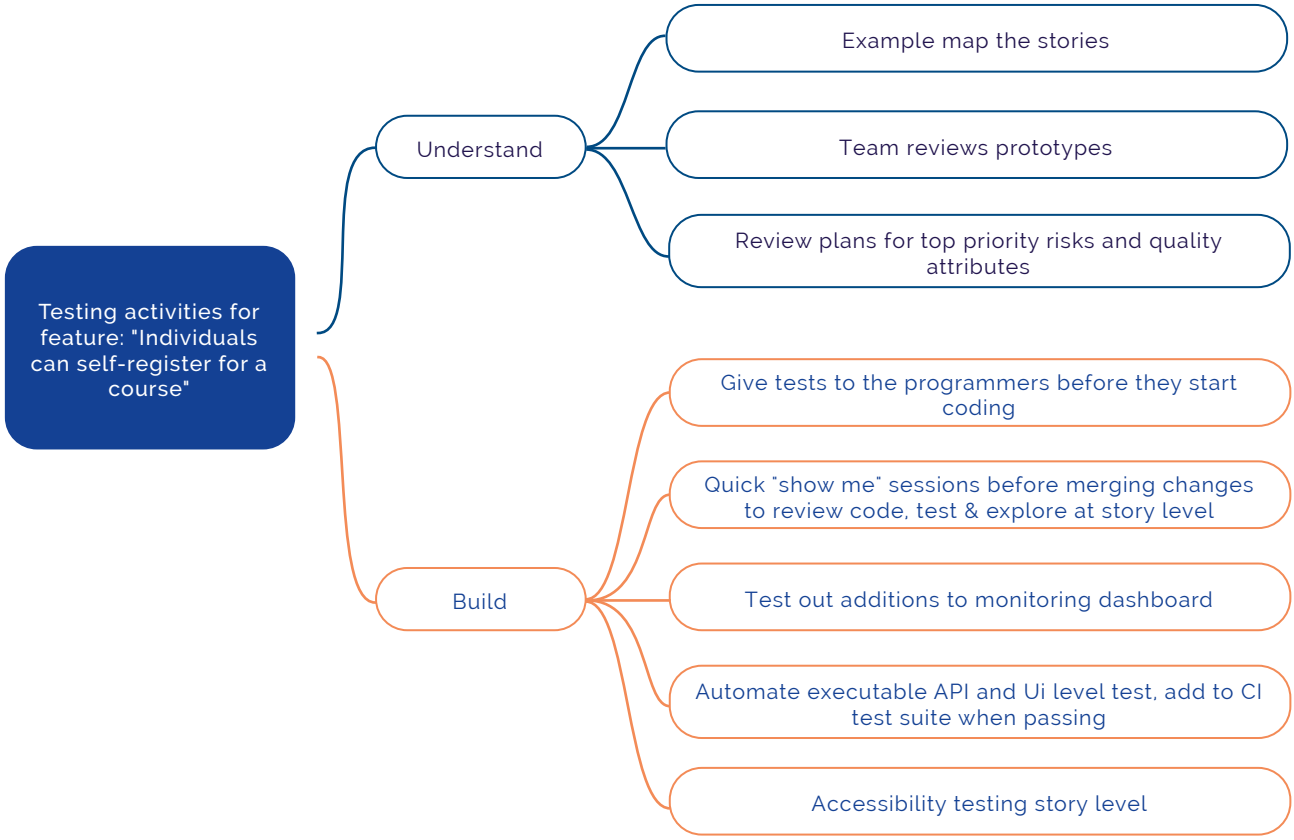
We mentioned earlier that your testing strategy document can be as simple as using the holistic testing model loop, with your own planned testing activities filled in at each stage. Another format we've used with good success is a mind map. Mind maps are a great way for teams to brainstorm together. They are easily updated, and can include

artifacts such as test results, screenshots, and links to other documents. Keep it visible.

Below is a sample of a mind map showing two stages – Understand and Build.

We've also used wikis to share test strategies across the organization. Whatever you decide to use, commit to reviewing it frequently and keep it up to date.

A holistic test strategy provides guardrails to help your team keep improving the quality built into your product. It eases daily work and ensures happy customers.



Lisa Crispin
Co-founder @ Agile Testing Fellowship, Inc.

Janet Gregory
Co-founder @ Agile Testing Fellowship, Inc.

TRIM Your Automated Tests

Author: Richard Bradshaw

Test Automation is not the silver bullet it was once portrayed as, a robot wizard capable of doing all your testing for you and replacing all your humans doing such work. Some magical sentient code capable of thought; deciding what should be automated, executing it, and creating bugs quicker than JIRAs own performance tests. Doing all this in sub fifteen minutes, with an ROI of one million percent. Reminds me of a test tool that I won't name which had an animation on its homepage bulldozing people (assumed QAs) out of an office building to show its promised value.

Having said that though, test automation is very much here, and rightly so, it's a wonderful thing to have on your team. It provides a constant workstream for all those testers. A flaky test here, a broken test there, a genuine bug here, a framework update there, and a green build to put into context. There's no shortage of work.

Some of you will have read this as humorous, but that isn't my intention at all, this is the automation trade off, a trade off we accept, if you are aware of it. You see, test automation, and in the context of this article, I mean automated test scripts, checks, automated testing, whatever term you prefer, can provide us with a huge amount of value, but that value comes at a cost that cannot be ignored. Quite the opposite, it has to be focused on, planned for, communicated and have a clear strategy to deal with this trade off, in order to benefit from all the good.

Flaky tests are inevitable, you simply cannot control all the complexities that come with software, yet alone writing more software to test that software! Accept they will happen, and have a clear plan of attack to reduce them, and tackle them when they happen. Tests will fail, we want them to, kind of the whole point of having them. But when they do fail, we need to quickly decipher what the failure is, a genuine bug, test framework, the test itself, or flakiness. Tests become outdated, a healthy sign of a good team, if your app is constantly changing and stores are shipping the tests need to go with them, which likely means some tests need updating, or even deleting. Yes, delete tests, all the time.

So, if we accept these things are going to happen, what can we do to maximise the value we are getting from our automated tests? Well that's where TRIM(S) comes in, a mnemonic created to help us create valuable automated tests that are supporting our wider testing efforts.

T - TARGETED

Our automated tests need to be targeted to a specific risk and automated on the lowest layer the testability allows. Your automated tests should be as small as possible, and ideally focused on a single risk, or a few at most. That risk should be explored to find the lowest layer in the application it can be mitigated. A lot of the time we automate scenarios or behaviours, but these are often very broad, to

maximise value we need small targeted tests. Take logging in as an example. Most teams have a login test on the UI. But what risk are you mitigating with such a test?
Does the username and password match? That the UI shows the correct form? That the right API is called? That a session ID is created? That a cookie is set? A lot happens in systems when we login, and if we have one single big test for such a behaviour, we significantly increase the risk of flakiness, and hugely increase the time it takes to debug such a test when it fails or needs maintenance. Instead, we need to break these tests into many small targeted tests, responsible for testing a single risk.

R - RELIABLE

To maximise their value, checks need to avoid flakiness, we need them to be deterministic. Are you testing your tests? You need to. As a minimum we should be reversing our assertions to make sure the tests fail, and you are testing what you intended to test. Ideally, we should be running tests several times locally before committing. Tests that aren't deterministic will steal valuable time from your team, and significantly reduce your mean time to feedback, which impacts your team's ability to make decisions about releasing.

I - INFORMATIVE

Passing and failing checks need to provide as much information as possible to aid exploration. A failing test is an invitation to explore, and that exploration will be significantly enhanced with key information from your tests. The more pieces of the puzzle your tests can provide, the quicker you'll detect the problem, and keep that mean time to feedback down. Think things like good test naming, is it clear what the test is doing so the engineer can match the code to the tests intent. It's the assertion clear and descriptive. Logging, have your tests provide a much detail as possible to what's happened during the test. Artefacts such as screenshots, videos, JSON dumps, snapshots, files that are going to speed up your investigation. Having the robots create these things for you, so everything you need will be right there.

M - MAINTAINABLE

Automated checks are subject to constant change so we need a high level of maintainability. We love to talk about this as an industry, and rightly so it's important. Some people view tests as living documentation, so when new features are added, old features removed or bugs are fixed, our tests also need to come on that journey with us. Therefore the easier we can make that effort, the faster our tests will be running again, and the lower our mean time to feedback will be. If your code looks like some Christmas lights that have been thrown into a box along with ten other sets of lights, it's going to take you significantly longer to get that green light again.

S - SPEEDY

Creation, execution and maintenance need to be as fast as the testability allows to achieve rapid feedback loops. I've mentioned Mean Time To Feedback (MTTR) several times in this article, let me explain it. I view automated tests as a mechanism for rapidly retrieving information about the system under test. Information that is critical for helping the team make decisions about the quality of the code/product. Decisions which will impact releasing, our customers and usually the company's bottom line.

Therefore, it's crucial we are getting that information as quickly as possible and that means we need to ensure creation, execution and maintenance are as streamlined as possible. Those three pillars need to work in parallel, one without the others is not enough. If you focus on creating TRIM automated tests you'll definitely get there. If you aren't there right now, consider using TRIMs as an heuristic against your current test automation and identify some areas you can improve on and reduce that all important MTTF.

Richard Bradshaw
Senior Architect, Quality Engineering
@ Slalom Build

The Importance of Feedback

Authors: Nicola Lindgren and Vernon Richards

This is an excerpt from the upcoming follow-up to the book "Starting Your Software Testing Career by Nicola Lindgren. The new book will be co-authored by Nicola and Vernon Richards.

We're talking about anything, from personal and professional relationships to careers and self-esteem. And we're not just talking about "negative" feedback either! Learning how to give effective feedback hasn't just helped me become a better teammate. It's helped me become a better person all-round. Still not sold?

Feedback has the power to build and destroy.

Here are 5 reasons we should all learn to give good feedback.

1. **Improves Communication:** Instead of giving vague or harmful feedback, you can provide actionable, insightful feedback.
2. **Builds relationships:** Providing constructive feedback and avoiding nitpicking contributes to a psychologically safe environment.
3. **Enhances learning:** A culture of effective feedback drives learning by helping people understand what they did, in what situation and what the impact was.
4. **Drives success:** Environments where it's safe to fail allow people to experiment because the consequence of something "not working" is learning.
5. **Boosts morale:** Learning why/how something works increases people's sense of progress.

Great but so what?

"Seriously Nicola and Vernon, all I need to do is report bugs! That's the only time I give anyone any feedback."
Oh yeah? What about...

- When you're collaborating on a story, figuring out how to implement and test the feature, and someone says something that is?
- ...or an amazing one!
- During a "lessons learned" session after a major release?
- In a retrospective. Especially when discussing what the team could do better next time.
- When you're part of the on-call team in the middle of a serious production incident
- And... bugs of course!

As you can see, there are endless opportunities for you to give feedback as a Tester.

How To Give Feedback Common Themes

There are LOTS of feedback frameworks and models. Some have 3 steps, some have 5. Some focus on actions, and some focus on relationships. Regardless of the specifics of each one or what kind of structure they use, they all share some common themes.

THEY GET SPECIFIC:
Instead of a hypothetical scenario or talking around an issue, use an example of something that happened.

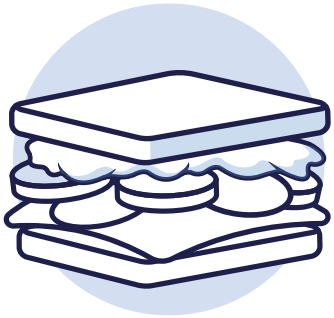
THEY DESCRIBE THE CONTEXT OR SITUATION:
When did this happen, who was there, why were those people present, what was said, etc. Set the scene for where and when the event happened.
They focus on objectivity, not subjectivity: It's the difference between "Bill shouted at Ijeoma" and "Bill was angry with Ijeoma". At first, stick to the facts and avoid interpretation. There'll be time for that later.

THEY SHARE THE IMPACT OF THE BEHAVIOUR:
What is it, and why do you believe it's so important?

In the next section, we'll share a few models we like to use to give feedback.

Useful Models

There are a bunch of models and frameworks for giving feedback out there!
So many, that if we listed them all we'd be here all year! Instead we'll list some of our favourites and a wild card thrown in for good measure.

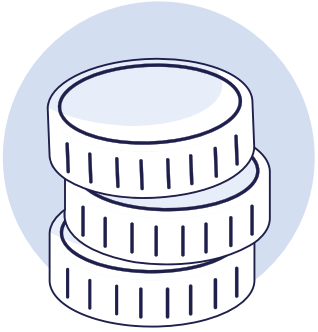


SBI (SITUATION - BEHAVIOUR - IMPACT)

I learned this one from an excellent Dan North talk called "How To Make a Sandwich", which you should be able to find on YouTube (recommended viewing!).
Situation: Describe the specific situation where the behaviour occurred.
Behaviour: Detail the actual, observable behaviour without interpretation or judgement.
Impact: Explain the impact of the behaviour and how it affected you, the team, or the project.

STAR (SITUATION - TASK - ACTION - RESULT)

I first heard about this method in the context of answering interview questions. However, it's also great for feedback in general!
Situation: Set the context for the story.
Task: Describe the task and the challenge involved.
Action: Explain the actions taken.
Result: Share the outcomes of those actions.



COIN (CONTEXT - OBSERVATION - IMPACT - NEXT STEPS)

This one is great for keeping the conversation constructive.
Context: Explain when and where the observed behaviour occurred.
Observation: State factual observations, not interpretations.
Impact: Share the effect of the behaviour.
Next Steps: Discuss actions or changes needed for the future.

These models are great, but they aren't a silver bullet. Next, we'll share some things to consider whenever you're giving feedback.

THINGS TO CONSIDER

We've explained the importance of feedback, given you some examples when you might give and provided some models you can use.

So you're all set right? Well not quite!

There are still some factors to consider when it comes to giving feedback.

Here are 5 things to consider when you're giving feedback

1

How's your state of mind? Giving feedback can take a lot of energy and focus, especially when practising it. So make sure you're in a calm and composed state of and not angry, frustrated or overly emotional (we aren't talking about being Mr Spock though!).

2

When is a good time? Try not to ambush people on their way to lunch or an important meeting! Choose an appropriate time and setting so people don't feel caught off guard.

3

Where are you giving this feedback? In our experience, people are more comfortable getting feedback in a 1 to 1 situation. And yes, that includes positive feedback more often than you might think!

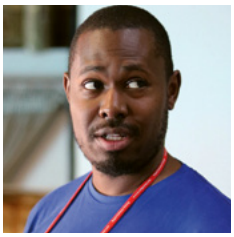
4

How clear are your intentions? People tend to judge others by their actions but themselves by their intentions. Try to avoid this mistake by making the intentions behind your feedback clear.

5

How could you be misinterpreted? This is bigger than word choice. Many factors affect how you'll be understood (or not!). Cultural differences, non-verbal cues, and our own biases have a huge impact on how we communicate.

Now. We've talked a lot about giving feedback. But what about receiving it? That's what the next section is about! But you'll have to grab the book when it comes out to read that part.



Vernon Richards
Senior Expert Quality Engineer
@ Ada Health

Nicola Lindgren
Platform Manager
@ IKEA

Photo by unicef on Unsplash

The Importance of Accessibility in the Modern Digital World

Author: Laveena Ramchandani

In the digital age, where technology permeates almost every aspect of our lives, accessibility has become a critical consideration for developing inclusive software and web applications. Accessibility ensures that all users, including those with disabilities, can access and use digital products effectively. As we advance into an era of heightened digital integration, understanding and implementing accessibility is not just a regulatory requirement but a fundamental aspect of creating equitable and user-friendly technology.

The Importance of Accessibility

1 Promoting Inclusivity and Equality

Accessibility in digital design is essential for inclusivity and equality. The World Health Organization estimates that over one billion people worldwide live with some form of disability. For these individuals, barriers in digital environments can hinder their ability to perform everyday tasks, access information, and participate in online communities. By prioritizing accessibility, developers and organizations ensure that digital content is available to everyone, regardless of their physical or cognitive abilities.

2 Enhancing User Experience

Accessibility is not just about compliance; it's about enhancing the overall user experience. Well-designed accessible features often lead to improved usability for all users. For example, captions on videos not only assist those who are deaf or hard of hearing but also benefit users who are in noisy environments or prefer to consume content without sound. Similarly, clear and consistent navigation aids all users in finding information more efficiently.

3 Legal and Regulatory Compliance

Many regions have enacted legislation requiring digital accessibility. For instance, the Americans with Disabilities Act (ADA) in the United States and the Equality Act 2010 in the United Kingdom mandate that digital services must be accessible to individuals with disabilities. Non-compliance can result in legal repercussions, including lawsuits and financial penalties. By integrating accessibility into your development process, you mitigate these risks and demonstrate a commitment to legal and ethical standards.

4 Expanding Market Reach

Accessibility can significantly broaden your market reach. By making digital products usable for people with disabilities, organizations tap into a previously underserved segment of the market. This inclusive approach can also enhance brand reputation and loyalty. Companies that are seen as champions of accessibility often receive positive attention and respect from the public, which can translate into increased customer engagement and business growth.

"Accessibility ensures that all users, including those with disabilities, can access and use digital products effectively."

The European Accessibility Act (EAA) 2025

The European Accessibility Act (EAA) 2025 is a significant piece of legislation aimed at enhancing accessibility across the European Union. Set to be fully implemented by 2025, the EAA seeks to improve the accessibility of digital and physical environments for individuals with disabilities.

1 Key Provisions of the EAA

The EAA mandates that various sectors, including digital services, transport, and public services, must meet accessibility standards. Specific requirements include:

Digital Accessibility: Websites, mobile applications, and e-commerce platforms must be designed to be accessible to individuals with disabilities. This includes compliance with established web accessibility standards such as the Web Content Accessibility Guidelines (WCAG).

Public Sector: Government websites and services must adhere to accessibility standards, ensuring that public information and services are accessible to all citizens.

Transportation and Public Services: Public transport services, including ticketing and scheduling systems, must be accessible to individuals with disabilities.

2 Impact on Businesses

For businesses operating within the EU, compliance with the EAA will be essential. The act provides a clear framework for accessibility requirements, which helps organizations understand and implement necessary changes. Businesses that proactively adopt these standards will not only meet regulatory obligations but also enhance their reputation and market reach.

3 Preparing for EAA Compliance

Organizations should begin preparing for EAA compliance by conducting accessibility audits of their digital assets, training staff on accessibility best practices, and integrating accessibility into their design and development processes. This proactive approach will ensure a smoother transition and minimize disruptions as the EAA comes into effect.

Incorporating Accessibility into Test Strategy

Incorporating accessibility into your test strategy is essential for creating inclusive digital products. Here's why accessibility should be a key component of your testing approach:

1 Early Detection of Accessibility Issues

Integrating accessibility testing early in the development process helps identify and address issues before they become costly to fix. By incorporating accessibility checks into the initial design and development phases, teams can ensure that potential barriers are identified and resolved promptly.

2 Comprehensive Test Coverage

Accessibility testing should be an integral part of your overall test strategy to ensure comprehensive coverage. This includes automated testing tools, manual testing by experts, and user testing with individuals who have disabilities. Automated tools can quickly identify common accessibility issues, while manual and user testing provide deeper insights into real-world usability and potential barriers.

3 Enhancing Product Usability

Accessibility testing improves the overall usability of digital products. By ensuring that your product is accessible to users with disabilities, you enhance its usability for all users. Features like keyboard navigation, screen reader compatibility, and customizable text sizes benefit not only individuals with disabilities but also users in different contexts and environments.

4 Fostering a Culture of Inclusion

Incorporating accessibility into your test strategy fosters a culture of inclusion within your organization. It demonstrates a commitment to equitable design and encourages all team members to consider the needs of diverse users. This cultural shift can lead to more innovative and empathetic design solutions that address the needs of a broader audience.

5 Meeting Regulatory Requirements

As accessibility regulations and standards become more stringent, integrating accessibility into your test strategy helps ensure compliance with legal requirements. This proactive approach reduces the risk of non-compliance and potential legal challenges, while also positioning your organization as a leader in accessibility.

6 Building Trust and Reputation

Organizations that prioritize accessibility build trust and a positive reputation among users and stakeholders. Demonstrating a commitment to inclusive design and accessibility can enhance customer loyalty, attract a diverse user base, and differentiate your brand in a competitive market.

In the modern digital world, accessibility is not merely a regulatory requirement but a fundamental aspect of creating equitable and user-friendly technology. By understanding and implementing accessibility, organizations can promote inclusivity, enhance user experience, ensure legal compliance, and expand market reach.

With the European Accessibility Act (EAA) 2025 on the horizon, businesses must prepare for the evolving accessibility landscape. Incorporating accessibility into your test strategy is crucial for identifying and addressing issues early, ensuring comprehensive coverage, and fostering a culture of inclusion.

By prioritizing accessibility, organizations not only meet regulatory requirements but also contribute to a more inclusive digital world where all users can participate fully and equitably. Embracing accessibility is a step towards creating technology that serves everyone and reflects the values of diversity and inclusion that are essential in today's society.

Laveena Ramchandani
Quality Engineering Manager
@ EasyJet

brightest

Reach Your Potential with Internationally Recognized Certifications

Unlock New Opportunities

At Brightest, we offer a comprehensive portfolio of globally recognized IT certifications, designed to validate skills in diverse fields such as software testing, software architecture, project management, and software sizing.

ISTQB International Software Testing Qualifications Board

IFPUG International Function Point Users Group

witpm Worth IT Project Management

iSAQB International Software Architecture Qualification Board

TMMi FOUNDATION

UNITED Certifications by brightest

SeU Selenium United

UXU User Experience UNITED

DOu DevOps united

API-U API UNITED

AiU Artificial Intelligence United

PtU Performance Testing United

CU Cloud United

DESIGN SPRINT alliance

AU Agile United

DAU Data & Analytics UNITED

Your Exam, Your Way: Flexibility Made Easy!

Our service provides ultimate flexibility—choose your exam date, time, format (online or at a test center), and preferred language. With our website available in 12 languages, we make the process accessible worldwide. Alongside your official certificate, you'll receive a digital badge from the market leader Credly to showcase your achievement on any platform.

Our Awesome Team!

The Brightest team is fun, versatile, and dedicated to your success! With 11 languages spoken, we offer attentive, multilingual support for a smooth, pleasant exam experience worldwide. We're here to help you #ReachYourPotential!

www.brightest.org info@brightest.org

Photo by Ian Schneider on Unsplash

The Road Less Taken

We're often told to become T-shaped testers but I'd rather be a "me" shaped tester. Following my passions and becoming a unicorn.

Author: Jenna Charlton

When Jose asked me to write an article for Agile Testing Days I was overwhelmed by the possibilities. I rarely get to write about what's on my heart and what I truly believe. As I considered my options, I couldn't get the word journey out of my mind. Having been recently laid off at the time of writing this, I was feeling adrift and a little scared. So many job postings were for automators who develop frameworks, or test managers who could contribute to the organization's automation project. I can't do those things, my journey and my passions have taken me in a different direction. Then, while at a concert, one of my favorite songs reminded me of the value in the road less traveled .

"I could have lived with my Gods as a Persian prince,
I could've played safe, But in the end the journey's brought Joys that outweigh the pain."
– Frank Turner, Journey of the Magi

Everyone's testing journey into testing is different. Some of us take a direct path from a university or bootcamp right into a testing career. For others, myself included, it's far less linear. We can find ourselves in a testing role after working on the business side at an organization or customer support, some of us even move from development to testing. No matter how you become a tester, what matters is your journey once you've become one.

YOU ARE THE AUTHOR OF YOUR CAREER STORY

You are the hero and main character of your career journey. I've often heard people liken professional development to a choose your own adventure story. However, I find that comparison to be far too restrictive. While there are multiple potential endings in a choose your own adventure book, the outcomes are predefined and you're not the author. The most empowering and terrifying thing you

can do in your career is own your agency and self determination.

Once you fully embrace your role as the owner of your story, the question becomes what path to explore. I have taken an experimental approach to my career. My skill set is wide and represents what I'm most passionate about. I was challenged many years ago to put words to what testing is to me. Not to what testing has given me (testing has given me a lot), what it represents, almost like a personal mission statement. For me, testing is about being of service to the user, my peers, and the business. Everything I've invested time and effort into learning has been in service to this mission. I began learning accessibility, user experience, developed skills as a trainer and speaker, and even became a researcher into the psychology and experience of work to better serve users and my fellow testers.

The best thing that comes from being the author of your story is your ability to change course. I have attempted to become a competent coder many times in my career. The rhetoric that tells everyone to become T-shaped largely drove me to revisit this path multiple times. However every time I've attempted to learn I've turned around and hiked back to what I'm passionate about. Most of us spend more time working than we do with our loved ones every week. I can't bear the thought of spending that time doing work I don't enjoy and isn't fulfilling.

PASSION AND A PARADOX

I am passionate about the research I'm doing and delivering a delightful user experience. For me, work only feels fulfilling if I have a deep affinity for the work I'm doing. However, passion is not required for success. If your career journey is that your job provides a paycheck and that is enough, consider this your thumbs up from the universe that your journey is valid. Not everyone is going to have passion for testing or work in general and that is okay. Frankly more career influencers need to embrace this. For those who like me need to feel deeply invested in and connected to the work they're doing to feel successful, finding what fulfills you is critical.

I was once asked by a friend who felt stuck and unmotivated how I discovered the work I love doing. The misconception is that we're always going to love the work we do, which is rarely the case. Instead, I had to identify what matters to me most in the kind of work I do. I've turned this process into an exercise for myself. I start by listing all of the words that relate to the kind of work I want to be my day to day focus. Then, I step through a process of crossing off words that are "work I enjoy" till I have a list of 5 "excites me" words. My 5 words are:

- Enablement
- Innovate
- Leadership
- Coaching
- Consultative

Once you unpack what truly motivates you, it becomes much easier to chart a course for yourself to follow.

FINAL THOUGHTS

After you find what matters to you most, connect with people who are already doing the kind of work you want to be doing. Look for opportunities to shadow, find a mentor, and people willing to sponsor you. Most importantly, reach back and be a ladder to those coming up behind you. Give back at least as much as you were given.

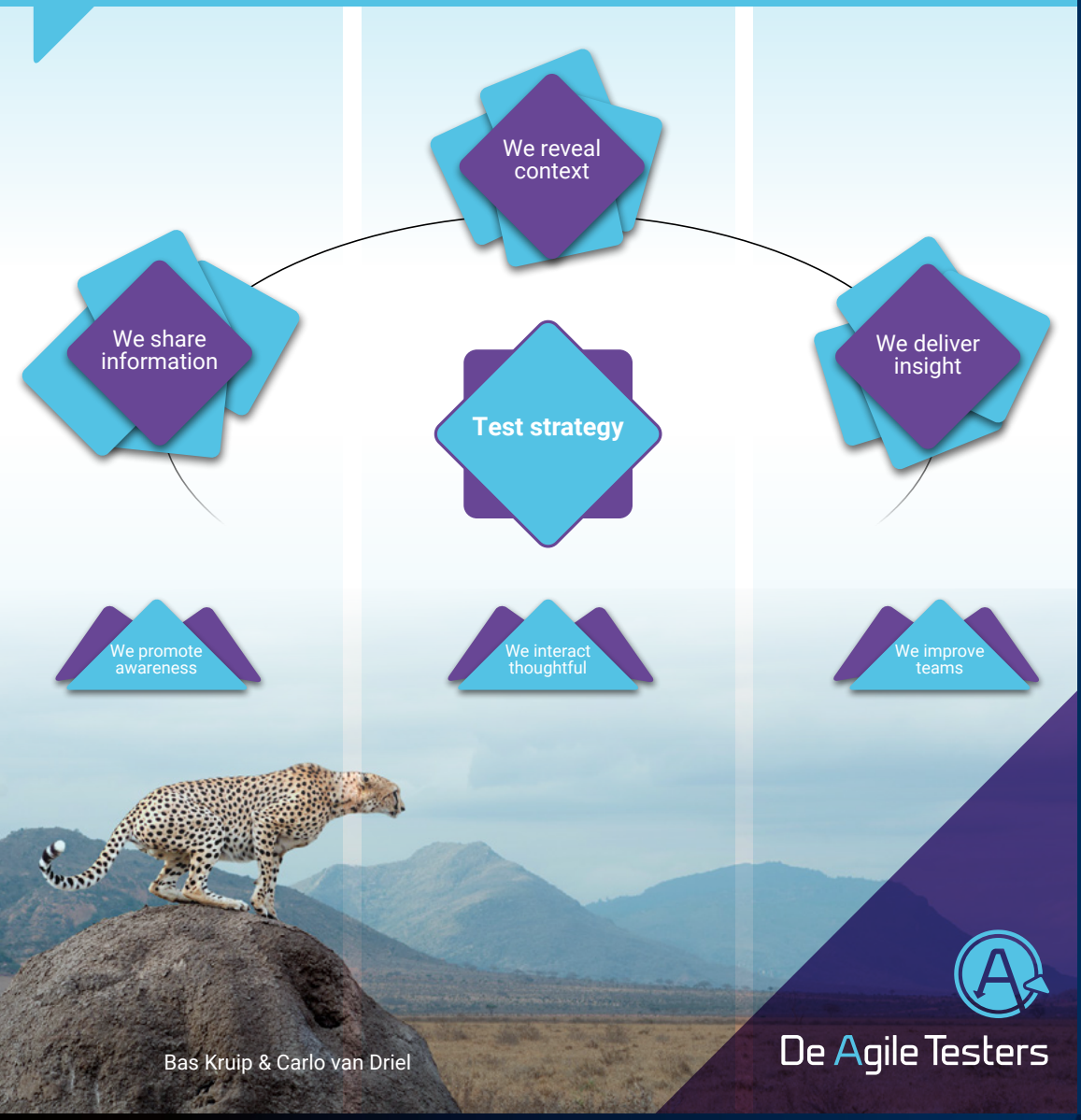
We spend a disproportionate amount of our time working and thinking about work in comparison to the time we spend with family, friends, and on our hobbies. While you don't have to have passion for work, you do need to be excited and motivated by something. Devoting time to the people and hobbies you love creates needed balance, because your happiness is even more important than your paycheck.

Life is about love, last minutes and lost evenings,
About fire in our bellies and furtive little feelings,
And the aching amplitudes that set our needles all a-flickering,
And help us with remembering that the only thing that's left to do is live.
- Frank Turner, I knew Prufrock Before He Got Famous

Jenna Charlton
Testing leader,
test strategy expert

FAST

Flexible Approach in Software Testing



Bas Kruip & Carlo van Driel

De Agile Testers

trendig

your leading technology services provider

Product Vision
Design Sprints

Consulting, Coaching, Training & Expertise Integration for

Agile Testing Days
Community Meetups

Agile Quality Practices

Agile Transformation & Scaling Agile

Requirements & Backlog Refinement

Software Testing (Automation, Mobile, Performance, Security)

Artificial Intelligence (Applications, Data Services, Testing)

trendig.com