INTERVIEW PART I

In the community of thinking testers, Mike Kelly is a familiar name. He is the creator of FCC CUTS VIDS heuristic, popularly known as Touring Heuristic.

Currently, Mike is looking after DeveloperTown as its Managing Partner among other interesting things he does.

Interviewing Mike has been on our wish-list and I'm glad that we could finally do it. Special thanks to Dirk for pairing up with me on this task.

Mike has provided insightful answers to our questions and for the benefit of our readers, we are publishing his interview in two parts rather than editing/shortening the answers to fit in one single issue.

I'm sure that you'll like what we discussed.

- Lalitkumar Bhamare

Over A Cup of Tea

with Mike Kelly

It's an honor talking with you today, Mike. We would like to know about your testing journey in short.

Thank you. I'm humbled that you'd ask to interview me. I don't do a lot of hands on software testing work any longer, but still view my time developing the skills of a software tester as critical to my success.

In total, I worked as a software-testing consultant for around ten years. In that time I contributed to two books, wrote hundreds of articles, spoke at numerous conferences, and helped launch the Association for Software Testing. My passions were always for test automation, performance testing, and exploratory testing. And early in my career I was blessed with access and friendships with the brightest minds in software testing - particularly James Bach, Cem Kaner, Rob Sabourin, Scott Barber, and everyone else in the LAWST community.

Today I'm the managing partner at DeveloperTown – a software-consulting firm. I'm the founder of Tenant Tracker – a commercial real estate SaaS product. I'm an investor and advisor in numerous startups. And on weekends I'm hobby farmer.

Among other cool things, your name is associated with FCC CUTS VIDS, popularly known as Touring Heuristic. We are curious to know if there is any story behind finding it.

FCC CUTS VIDS emerged out of a weekend spent with James Bach. James was trying to get me to change the way I thought about dissecting a product. He had mentioned the idea of touring a product several times that weekend, and had challenged me with numerous exercises around different ways to identify meaningful dimensions, factors, systems, or components in a product. It was shortly after that weekend that I started to write down my ideas based on a lot of the material we had covered. I view FCC CUTS VIDS as an extension to his work.

After coming up with the heuristic, I used it all the time. It became my default way to start my test planning process for an application or feature. And early in a project it allowed me to quickly identify where I had questions around the various aspects of test coverage and risk.

Considering changing technologies and complexities of software in general, what would you like to add in existing considerations under FCC CUTS VIDS?

Hah. I wouldn't add anything, because then I wouldn't be able to remember it as easily. That's the downside of mnemonics – once you've memorized them they don't like to change.

Based on my more recent experiences, I think I'd try to figure out a way to add two things:

1) Money Tour (or Value Tour): How does this product make money? Or – said in another more general way – how does it deliver value back to the people who produced it? Can you quickly identify all the economic factors either explicit or implicit to the software and the underlying business model behind it? Based on how the product is monetized (or however value is captured), what are the necessary conditions for that transaction (or transactions) to take place? How and when does a user complete a financial transaction? I think this tour helps get at business risk in a way none of the existing tours address.

2) Operations Tour: I'm not going to explain this one well, but I'll try. There are a handful of common technologies/solutions that we integrated with products today that interact with our customers. Examples include Zendesk, Olark, and Okta – simple and focused solutions that touch our customer and their experience with our product(s). There are also a handful of products that we use behind the scenes to monitor and troubleshoot real-time customer issues like New Relic, Google Analytics, or Slunk. They don't "touch" the customer, but we use them to craft better experiences over time. I've come to rely on data from these types of operations-focused tools to help me build my model of what I'm testing and where risk might be. And it also tells me something if we aren't using tools like these to manage our user experience.

You wrote a chapter in "How to Reduce the Cost of Software Testing" which is popular book in testing circle. How was your experience?

My experience working on the book was great, but I feel like I cheated a bit. I was able to repurpose some materials from a series of articles I had written on exploratory testing. So while I refreshed the material and changed it a bit to fit the theme of the book, I had a solid down payment on a first draft before I even started. The editorial team for the book was easy to work with and clear on deadlines. And the reviewers for my chapter were gentile and insightful. I'd do it again without hesitation.

Currently you are focused full time on DeveloperTown. We are curious to know about your experience with DT about building quality in.

Over the last six years we've been working with clients (both single person startups and Fortune 100 companies) to launch brand new SaaS and mobile products. During that period, we also ate our own dog-food and founded our own SaaS startup (www.tenanttracker.net). As a general rule, at DT and at TT, we work in two-week iterations and try to release software as quickly as we possibly can post iteration. Most teams are somewhere between three and ten people in size, and have some mix of design, development, testing, project management, and marketing. Most of us often end up doing some amount of multi-project multitasking despite our best efforts.

On most projects we write "less than perfect" stories and instead try to rely on frequent communication and a high-level of ownership among team members to work through it. Sometimes we have great product owners (clients) who are laser focused and knowledgeable about the product and the market. Sometimes our product owners (clients) are… less than awesome. All of our clients are trying to hit crazy deadlines with about 50% of the money they actually need to do it "the right way."

(If you're one of our clients, clearly you're in the awesome category. I'm not talking about you above.)

We aren't always perfect with standups. We don't always get our retrospectives done. And (depending on the client) some of our planning sessions are more collaborative than others. We have a guiding process, but trust the team to tailor that process on a project-by-project basis.

In our context, testers have multiple projects, crazy deadlines, little documentation, and a lot of responsibility. They also have a lot of trust, we allow them the freedom to select the best tools for the job, and they have a team that (all things being equal) wants to support them as best they can.

We accept that while testing is a role (and it some cases a specialty), we also want it to be everyone's job: our project managers test, most of the time our developers test, most of the time our product owners test, and when we're lucky we can have early alpha/beta customers test. When we assign a tester to a project, we're typically looking for them to find the things that others will miss. We're looking for them to prioritize work based on risk, and to try to identify areas of coverage that other people on the project just aren't thinking of.

We need team members who:

- have great technical skills;
- are comfortable making decisions will little information;
- know how to ask great questions;
- take ownership of their work;
- don't have big egos;
- and are good at communicating with the rest of team.

New product launches – more than any other type of project – best illustrate the tradeoffs of product/market fit, market timing, budget, and technical complexity. Quality in an environment like this is delicate to balance. You're always trying to make the right tradeoff. Even for well-funded large corporate clients, we have to balance tradeoffs. Testers should be some of the best people on the team at identifying tradeoff opportunities and bringing those to the attention of the rest of the team.

You have rich experience with Agile development methodology. From your experience, what trends do you see coming in there that testers must prepare themselves for?

Agile can mean a lot of different things. At risk of avoiding the question, I want to point people back to my answer to the last question. If that sounds like agile to you, then we're doing agile. If not, then we're doing something else and you can ignore this answer.

That said, I've done testing in waterfall environments, RUP, agile, and complete chaos. I've never found that the core skills I've needed to be an effective tester on any of those projects has changed based on the process the team was using. Documentation was more important in waterfall, knowledge of the Rational tools and templates helped with RUP, and the ability to task out your work and estimate quickly will help you in agile – but fundamentally the testing is the same.

For me, test planning is about modeling, assessing risk, identifying coverage criteria. Testing is about designing and executing actual tests based on your models. And test reporting is about effectively communicating those results. None of those core tasks change based on the team's process. What changes are: how much time you have, how much documentation you need, and people's preferences for communication styles and tools?

For me, the trends that matter for software testers working on agile teams have less to do with testing, and more to do with how they can help the rest of the team as a generalist. I think agile testers are expected to be a bit more of a Swiss army knife. Not only do they need to be good testers, but also they need to be amateur developers, have exposure to analytics, need to know the basics of good UX, and need to be excellent researchers. On most agile teams I've worked on as a tester, we've been not only the "tester," but also the backstop for a number of other roles for the project. I think that's a natural extension of our broad focus as testers.

Do you think there will be any scope/demand for specialization?

I always think there will be space for a specialization. But I think what that specialist brings to the table will change over time, and based on the project.

Doing a lot of mobile apps? Your tester is going to have a ton of hardware. Doing an IoT project? Your tester will likely have some hardware/device background to draw on. Building a big data solution? Your tester will likely be able to rock some specialized mathematics, and will likely have some non-trivial performance testing experience etc.

I think that's the challenge. There's an expectation that testers need to simultaneously become generalists and specialists. They need to be able (and willing) to ask "How can I help?" While at the same time bringing some deep experience/skill to the team that others rely on. "I was running some battery-life tests, and it seems like we're still chewing up the battery on the device. Are we properly doing the handoff to the hardware h.264 codec? We hit that on a past project."

While I think that's the trend, I think first and foremost you just need to be a good tester. Being a good tester comes before everything else. If you can't do the core job, you can't paper over it just by being helpful or by having a hard-to-find one-off skill. Agile teams move too fast for underperformers to hide for long. Become a great tester first. Then worry about what else you can bring to the team. The trends will change.

That was Mike Kelly on his past and present work, his opinions about Agile. Join us in part two of this series where we continue to discuss some interesting aspects about tester's skills, other fields that fascinate Mike and some other cool topics.

Stay tuned for part two….

- Lalit and Dirk