Five Questions With Paul Carvalho

Written by Michael Hunter

04/22/08

Paul Carvalho is one of the many Canadians who are actively contributing to the Test community. You may have read his article on internationalization testing for Better Software, his article on hiring testers for Red Canary, or his blog wherein he divulges the secrets of his testing success.

After work Paul's children help him decompress by ensuring he has neither time nor need to focus on any task for more than a few minutes at a time. Once his kids go to bed Paul studies origami and volunteers at Ontario's Perimeter Institute for Theoretical Physics. And still he finds time to review books and university courses, and to contribute to the Software Engineering Body of Knowledge. (Send kudos or complaints, depending on your view of that corpus, to Paul's email above.)

Here is what Paul has to say:

DDJ: What was your first introduction to testing? What did that leave you thinking about the act and/or concept of testing?

PC: I have thought about this question over the years, and I don't believe my first testing job was my first introduction to testing. Rather, there were several different experiences leading up to that first testing job that helped me realize when I had finally found my niche.

In High School, I used an Atari 800XL to connect to Bulletin Board Systems by modem. This was the 1980's and before the Internet was publicly available. I learnt a lot about communication protocols and could even whistle 9600 baud! (That means that I could dial a number and trick a modem into thinking that I was a computer just by whistling into the phone. What a geek. Ha, ha.)

Occasionally I would download some software, a game or an application of some kind without any documentation, and I would spend several days playing with it to see what it could do and what I could figure out about it. It was fun, a game: randomly and methodically hitting keys on the keyboard to see what happened, using tools and editors to read parts of the code, and making notes along the way to keep track of what I learnt. I rarely spent more than a week on any particular program before deleting it and moving onto the next one. This was similar to the Exploratory Testing that I do more formally today.

Later I went to university and majored in Physics. I studied and practiced the Scientific Method and problem solving on a regular basis. I became pretty good at modeling and pattern detection to help me "see" solutions in my mind to assigned problems. Essentially, I became very good at General Systems Thinking, a very useful testing skill, although I didn't know that name at the time.

While I was still in school, I held several contract positions as a programmer, and eventually came to work at Delrina in their Technical Support department in the early 90's. When I started there were fewer than 20 people in the department, but it was the right place at the right time and our department grew by four times within a 6-month period as the company experienced tremendous growth and popularity.

At some point, we started encountering more problems in Support than the QA team could help us solve, so we created our own test group within the Support department to help us get some answers quickly while the QA team focused on testing the next release. My thorough efforts in this Support Test group got the attention of the QA Manager. He came up to me one day and offered me a job in the QA team for the following summer, which I accepted.

On my next school break, I returned to Delrina in the QA department and worked on two cool projects: the modem compatibility project, and testing a new Forms-based information management system. In the four months I worked there I tested hardware (modems), managed a network of computers running through a series of automated regression tests, observed real honest-to-goodness Usability Testing, gained my first exposure to test techniques, and helped write automated test scripts to test the next generation Forms system.

So what did that leave me thinking about the act or concept of testing? I thought it was cool. I thought it was fun. I thought about the diversity in how testing was done in this one company and how each of the different ways helped to make a better

quality product and experience for the end users. I reported bugs and they got fixed. I interacted with both Programmers and Tech Support people and enjoyed being the go-between. There was a "science" aspect to Testing as I drew upon my modeling and problem-solving skills from my school experiences. There was also an "art" to it as I drew upon my high school hacking hobby to feel my way through a system and find the information I needed.

I was bitten by the Testing bug and I liked it!

DDJ: What has most surprised you as you have learned about testing/in your experiences with testing?

PC: How complex it is to do a good job in testing. How diverse the range of skills are that you need to be successful. How important a role Psychology plays in software development. How many people vastly underestimate the effort required to do a good job of testing. How shockingly little most developers know about effective testing techniques. How little most educational institutions offer their students in the way of Testing instruction. How much the Testing body of knowledge still has to develop. How much debate there is over terminology and certifications. How hard it is to hire a good senior tester.

Testing has a solid basis in the Scientific Method and so it is open to many of the same theoretical, practical and philosophical debates that you encounter in Science. Things like: the nature of observation and bias, control and awareness of variables within your tests, the role of Statistics and Probability in test design and analysis, the Psychology of information presentation (i.e. for increased understanding), the variability in repetition of experiments, validity of assumptions, and so on.

The people who don't understand this generally do bad testing because their superficial view leads them to believe in simplified 'best practices.' Those of us who have a pretty good idea of what we're doing spend a lot of time trying to undo the damage caused by the processes, beliefs and expectations that sprout from these best practices.

I can't just pick one thing. Many things still surprise me.

DDJ: What is the most interesting bug you have seen?

PC: That's a tough one because I think many of them are interesting in their own way. I still have a very fresh attitude towards testing. Every day is a new opportunity to find an interesting bug. And I can't turn it off either. I see bugs everywhere - in the Grocery store, in newspapers and flyers, children's toys, restaurant menus, you name it. The world is a pretty buggy place.

Some bugs just appear to me by accident, some because I go looking for them, while others require lots of careful and detailed prep work and setup beforehand. For example, I once spent several days Performance Testing a web application to determine the maximum message throughput for a queuing component. I discovered that the application really supported 100 times fewer messages than expected. That was one of those "back to the drawing board" moments. Oops.

One of my favourite accidental bugs happened about a decade ago, when I worked for a company that made commercial math software. The software essentially had two parts: the math engine and the GUI. The company's corporate logo included an interesting multi-coloured 3-D graph representing a calculus function. I remember seeking out the mathematical function that produced the corporate logo and adding it to one of our regression test suites. In one development build, I discovered that the math engine had changed in a way that caused the math function to produce a different graph! I think the bug report I created had a title like "We can no longer produce our Corporate Logo with our own software!" They fixed it right away. We all had a good laugh over that one for many weeks afterwards.

One of the strangest bugs I remember happened when I was working in the Tech Support department at Delrina back in the early 90's. We were supporting Fax modem software in early versions of DOS and MS Windows. There was one customer who had bought our software and a supported fax modem but he just couldn't get the two to work together. My colleague who took the call tried everything. He checked and changed the computer's BIOS (this was before Plug-and-Play), added the necessary changes to the Windows 3.x environment, and tried a variety of modem initialization and configuration strings. Nothing worked. He felt bad but had to tell the customer that we just couldn't figure out why the software and hardware didn't work together. Some time later that same customer sent us a fax using our software with the message: "My computer got hit by a power surge during a lightning storm and the modem started working with your software! Thanks for your help!"

That fax was posted in our lunchroom for many, many months. That was funny. Weird too. Not a recommended fix though: modem doesn't work? Zap it with lightning! =D

DDJ: How would you describe your testing philosophy?

PC: I think a testing philosophy has several different aspects to it.

To begin with, I think that Testing is best understood in the context of the bigger picture of what is generally going on in Software Development projects. That is, software development is essentially the process of translating an idea from someone's head into computer instruction sets that perform some tasks useful for some purpose. Time and cost also factor in here, but people are the biggest part of the equation. So, if software development is a creative, people-based translation process, what is testing?

I believe that Testing is a way by which we empirically check the progress of the brain-to-code translation. If done well, Testing provides the feedback required to make the necessary adjustments along the way to ensure that we hit the target at the end of the project. If done poorly or not at all, the information may be insufficient or come too late to be useful in this process. Missing the target may have huge consequences. For example, people may lose their jobs, people may lose their finances, or people may die, to name a few.

How is testing done? It is simply done by shifting your perspective on the creative process, asking questions, making observations, and reporting findings to the decision makers responsible for the product release. I suppose "simply" may not be the best term here since each one of those steps require skill and knowledge in order for you to be effective and successful.

I believe that the Scientific Method and Statistics are cornerstones in effective testing approaches. I believe that you have to learn how to recognize a bug and it begins with how you think about the system and what you think the correct expected behaviour should be. Psychology also plays an important role in Testing. Testers are influenced everyday by many people and distracted by many things, and we need to be aware of the influences and biases that we bring to the job.

Because of the impossibility of complete testing, decision-making and good judgment skills are also important in Testing. Key project factors such as quality, time and cost need to be integral parts of the daily decision-making processes, whether you are trying to decide which features to test (or skip) or helping to decide when to ship the product. "Risk-Based Testing" is a popular approach that people may use to evaluate the importance of their testing effort based on the impact of an event and the probability of occurrence (back to Statistics here). To be useful, these decisions need to be made regularly and not just once per project.

Overall, I see testing as a creative, methodical, investigative, disciplined, fun, people-based way of helping to turn ideas into products. Testing is a service to the development organization, to the creative process of figuring out what should be in the product and how it should work. Good testers generate good, timely information for the stakeholders to use - in time for them to use it.

DDJ: What do you see as the biggest challenge for testers/the test discipline for the next five years?

PC: I've watched Agile Development methodologies slowly gain momentum over the last five years. I knew many development managers who were intrigued by the Agile Principles five years ago but were afraid to try the new ways of developing software. I have seen a shift happen in Development as more and more companies are adopting agile practices. That shift has yet to begin with Software Testing.

There's an approach called Exploratory Testing (ET) that has been around for a while but many people have been afraid to try it. Unfortunately, there appears to be much confusion over what ET really means and how to do it. As a result, there are many test teams still doing things the old way, relying upon their trusty test documentation and not realizing that they are making themselves obsolete.

This 'old way' that I am referring to is a documentation-based testing approach where someone creates documents with many, many test cases and then someone (else?) executes those test cases to generate test reports and other documentation. I suppose it keeps the Paper Industry prosperous.

One of the major problems I have with this approach is that it fundamentally follows a "Waterfall" model, and therefore is not compatible with Agile practices and principles. I have already seen some companies struggle to understand why their testers

aren't really helping on Agile development projects. Therefore, the quick conclusion must be that testers aren't required on Agile projects, right? Not so!

Exploratory Testing, when performed by skilled testers, may be a very effective complement for Agile projects. It is adaptable, responsive to change, effective and produces exactly the kind of information that developers and stakeholders need on the project.

So, over the next five years, I see several big challenges facing us.

For testers:

1. Testers who wish to remain useful will need to learn new techniques and new skills that will enable them to be more effective, adaptable and responsive to change in a fast-moving industry.
2. Exploratory Testing is going to gain popularity in understanding and usage. This is going to change how test teams work together. It will also change how outside people work and interact with test teams. People who are afraid of change should just get out now. ;)

For the test discipline:

1. More courses in Software Testing and good testing practices in context need to be created and integrated into Computer Science and Software Engineering programs at educational institutions everywhere.
2. New Test Management and communication frameworks need to be developed to help test teams work effectively in a diversity of project sizes. There are very few options available today to help you manage or communicate your ET efforts. More research and collaboration will be required to help the test discipline move forward in a good direction.
3. Most of the vendor Test Management tools are based upon the central idea of a "test case." When that concept becomes obsolete, vendors will have to redesign their tools to be useful to testers again.

In summary, the biggest challenges facing testers are in self-development and embracing new approaches that improve their effectiveness and usefulness. The biggest challenges facing the test discipline are in educating more people in useful testing knowledge and techniques, and in developing new tools and techniques to help us manage the evolving testing practices. Shouldn't be too hard, right? =)

[See my Table Of Contents post for more details about this interview series.]

Last Updated (04/21/08)