Learning to Lead: Making an Impact by Improving Yourself

Carol Brands
carolsuebrands@gmail.com

Final Revision
6/10/2015

“Change is hard,” is a common refrain when talking about leadership. It feels especially hard when working to create change without the benefit of an official leadership position. This was the situation I found myself in over the last few years in my current team. I was the second-newest member of a four person team of testers, collectively responsible for testing all of our company’s products, and I was the person on the team with the least experience as a tester. As a testing novice, my understanding of the effectiveness of my team’s methods and what other methods might exist was vague, but I had a hunch that there was more my team could be doing to develop into a stronger more efficient team.

Start with Personal Development

I wanted to become a better software tester, so like all students in search of a quick solution, I turned to Google. There were plenty of websites that claimed to be about software testing that all looked equally valid to my inexperienced eyes. I started looking for a few places that might bring these vast resources together, and one of the first that gained my attention was the blog feed on the Ministry of Test website.

One of the benefits of the blog feed was the exposure to a wide variety of other software testers and their thoughts. Everything from the relatively inexperienced tester writing their first blog post relating their favorite hobby with testing to the musings of highly skilled craftsmen seeking to advance the field. Among the posts that came through the feed was an entry from the Satisfice blog by James Bach titled “To The New Tester”. This is the blog that really exposed me to the next level of development.

In the blog, Mr. Bach encourages a participatory approach to development. Beyond simply reading, his advice included joining Twitter and participating in conversations there, participating in Weekend Testing, and practicing testing. These are all suggestions I took to heart. I found and added James Bach on Twitter, then begin looking through the people he followed, and the people they followed, until I had amassed a broad collection of viewpoints to follow and discuss ideas with on Twitter. I joined Weekend Testing Americas sessions. These are sessions generally conducted on Skype, where a facilitator and a group of peer learners gather to discuss a specific testing topic. I also discovered the Miagi-Do School of Testing, another collection of peers seeking to offer each other challenges to practice testing skills and develop the testing community. Most recently, I found Software Development 24/7, a group where software testers can discuss interesting topics, collaborate on writing, and share ideas about the day-to-day business of testing.

Learning New Ideas and Techniques

Rather than using these new-found resources for personal development to seek solutions to specific problems, I started by taking in a broad range of information to understand the world of testing that lay before me. As ideas that might work in my environment appeared, I took note and added them to my mental list of things to study and try in the future. Some of the ideas I have come across and begun evaluating are the use of heuristics, testing against personas, and the false comfort and value of a ‘test everything’ mindset. The ideas that have influenced my daily work the most so far are: Rethinking scripted testing and pair testing. While I have come across many other interesting ideas in my studies, these are the two that I felt could improve my own testing skills and my team’s effectiveness the most.

Rethinking Scripted Testing
My first experience with testing was as a technical support representative at a small company. We had a testing list that was written for us by a developer. It was written as a long checklist of things to examine with little direction, and it was intimidating. As I got more familiar with the program I was testing and the intent of the document, I became more comfortable with it, but I still felt it needed more detail. I wanted a full list of every behavior I should check, not a summarized list of features and potential failure points.

Later, when I arrived at my new job as a Software Tester, within the first week I was assigned “test cases”. I was overjoyed. “Test cases” is the phrase my company uses to refer to the test scripts that we execute to examine our products. Finally, someone was telling me exactly what to do and exactly what to look for as a result. This felt so much better than the vague test document I had worked with previously. Until it didn’t. I quickly found that the scripts I had been assigned were out of date. I constantly needed to question my co-workers to confirm the scripts were still correct. More often than not, they weren’t. The data the script depended on had changed or a change in a feature behavior had been made since the instructions were written, and the script was no longer meaningful.

When I began studying testing, I came across many resources that reinforced the failures I had seen in using test scripts as a method of performing and managing testwork. I first saw Paul Holland’s “Skills Based Testing” video through a blog post by Ben Austin reflecting on his Easter egg hunt analogy. In the video, Paul compared bug hunting to an Easter egg hunt, where someone might search for hidden eggs in various rooms of the house. He suggested that just as searching in the same locations repeatedly would not reveal new eggs, repeating the same tests over and over again would not reveal new defects. This resonated with me, because we had just completed a regression using test scripts, and we had defects escape the building. Our typical answer to this problem was ‘write a test script for every found defect to make sure it is covered in regression’, but this model of coverage had clearly failed for this particular regression.

Around the same time, I also began seeing alternative methods of modelling testing and test coverage, such as Katrina Clokie’s “How to create a visual test coverage model”. Discovering that alternative methods of explaining test coverage exist encouraged me to look for other ways to accomplish what we had been doing through scripted testing. If there are alternative ways to model coverage, then it seemed likely that there may be alternative ways to manage testing and discuss the testing status of a product. Exploring these alternatives, with their benefits and drawbacks, may help us improve our ability to test and share the results with others.

After discussing these ideas with other testers outside of my company, I started to bring up these ideas within my test team. Whenever we discuss team improvement, I suggest that we may be able to avoid some of the drawbacks of scripted testing by changing the way we write “test cases”. Some of my teammates are not yet convinced that scripted testing may be unnecessary or that there are better alternatives, but I continue to bring up the drawbacks of scripted testing whenever the conversation comes up. As a result of bringing it up regularly and pressing for exploration of other methods, the team agreed to make researching how we can write “more effective” test cases one of my goals this year, leaving room in the wording of the goal for me to include non-scripted testing methods. I hope to find more alternative methods of performing and managing our test work that avoid the drawbacks of using scripted test cases.

Pair Testing

As I continued to study testing, I came across the idea of pair testing, first on Twitter, then in blogs and in conversation with other testers. This idea was particularly appealing to me, because it seemed to address some specific practices that I hoped to improve on our team, especially knowledge sharing and performing non-scripted testing. As I learned more about how people are using the practice, it seemed that pair testing may provide a strong improvement on our training methods.

When I first arrived to my company as a tester, my training consisted primarily of reading documentation and executing test scripts. While following the scripts, most of the learning process came not from the scripts themselves, but from asking questions about outdated scripts and confirming that the scripts that could be successfully completed were correct. It was a painful process, because no one wants to take another coworker’s time needlessly, but I did not know enough to know if my questions were warranted. This method of training left me feeling helpless, but it seemed necessary at the time, since my more experienced co-workers were busy getting other testing done.

When discussing the idea of pair testing, a few bloggers mentioned that they often pair with new testers. I thought perhaps this meant that pair testing could be used as a training method. The experienced tester would be able to complete the assigned testing for the day, and the new tester would simultaneously witness the experienced tester’s methods, and learn the program functionality by using the product during testing. When questions arose, the experienced tester would be able to answer those questions immediately, so the waste of searching and waiting for someone to answer questions would be minimized.

This idea was very exciting, but we did not have any near future plans to hire a new tester, so when a new support person was hired, I immediately reached out to the Support Team supervisor to ask if I could try out pair testing as training. I explained that the support person would get the benefit of thorough training in the feature we would be testing (as well as learning about testing), and that it might benefit future training efforts for both the Support and Test teams. She agreed, and we began the next week. We completed two testing sessions, one without using scripts with a feature that I was less familiar with, and one using scripts for a feature that I am very familiar with.

The two sessions revealed that differing methods of using pair testing as training gave different results. In the first session, I drove the session, but clearly explained how the feature worked and what we were testing. The new support person began feeling comfortable with the feature we tested very quickly. In addition to training, this session had some unexpected benefits. His participation revealed a few bugs that I had not noticed when working with the feature on my own. Since the program was still new to him, he was able to spot a few inconsistent behaviors that I had been conditioned to ignore over time. His questions revealed that a few feature behaviors I had been taking for granted were more questionable than I realized, so I learned the feature even more thoroughly by seeking answers to his questions. This session felt very successful, since in addition to completing the testing and training the new support person, we both gained product knowledge and found more bugs than we would have by following scripts independently.

In the second session, he drove the session, using the test scripts as a guideline for what we were testing. This session was more difficult and less engaging. I was still able to describe the feature and what we were testing, but the scripts made the session feel like an exercise in button-clicking. The support person mentioned that he thought this was boring. This session felt much less successful than the previous session, since we didn't extend our learning beyond the scripts, didn't experiment as much, and didn't reveal many defects. In retrospect, I might have followed the scripts, but I would have directed him through the steps without calling attention to them. Instead, we would have talked about what we were doing and why, more like the previous session. In this way we may have captured more of the engaged learning and exploratory testing of the first session.

Since completing the training experiment, more opportunities for pair testing have become clear to me. I was able to set up a small group testing session for a new product with one of our testers, a marketer, and a product manager who specializes in design. This provided a wealth of benefits. The marketer was able to get his hands on the product and provide the tester with feedback based on his user research. The designer was able to help us identify usability defects. The tester had recently been struggling to understand the value of the new product, and was able to re-engage with it while working with the marketer and product manager as customer proxies. This experience reinforced how much value pair testing had to offer our team.

Moving the Team Forward

The brief descriptions above quickly gloss over the time and effort it took to move from learning about testing to bringing those new ideas to my team. Making those changes required more than just an understanding of the methods being introduced. Prior to introducing change, a foundation of relationships needed to be set and the ideas needed to be introduced to the team through those relationships.

Turning Relationships into Allies

The relationships that will become important allies for making change are sometimes obvious. When I joined the test team, it consisted of three testers reporting to a manager with no testing experience. I found one of the other testers in particular easy to talk to. When I was just starting out, I would sometimes talk to him about ideas I'd had but was afraid to mention during team meetings. If he found the idea particularly good, he would mention it to the group, making sure to give me credit for the statement. This behavior made him a clear ally as someone I could trust and confer with when I was unsure whether an idea was ready to present to the group. Now that I have more experience, I often discuss new ideas with him before bringing them to my team on my own.

Other relationships are less obviously beneficial for making changes. When I first arrived in my current team, I had come from a technical support position. As a result, I often spoke to our support staff when I had questions about the product or wanted to know more about our customer's usage. I also offered my services to Support staff often, helping them to examine user-reported bugs and answering questions about new features that had recently come through testing. The Support Lead noticed and encouraged my interaction with Support.

By creating relationships both inside and outside of my team, I have been better able to try new things that will help my team improve. Through my relationship with Support, I was able to introduce pair testing as a potential training method that may improve both our teams. Through my relationship with my team, I was able to further examine pair testing as a testing method for experienced testers to harness the knowledge other teams in the company.

Participate in Hallway Discussions
Part of developing these office relationships was talking to my co-workers often outside of formal meetings. Hallway and short office discussions were a great way to make inroads to bringing some of the new ideas I had been learning about to the team.

One particularly important conversation occurred after a company meeting. A product manager had mentioned in the meeting that he was congratulated by his peers on the design of his product. The test team had been working on a new, different product that was experiencing some design weaknesses, so I saw an opportunity for pair testing that might prove its value beyond training. After the meeting, I approached the product manager casually, congratulating him on his success. By taking the opportunity to start a conversation, I was able to introduce him to the topic of my team's trouble with testing the design of the new product. I talked a little about how including him in a pair testing session might give us some insight on the product we were working on and asked if he was interested. He thought it was a good idea and was happy to share his insights with us.

After this conversation, I spoke to the tester that I often use as a sounding board. He mentioned that he was frustrated with some of the current design decisions on our new product. This was the perfect opportunity to bring up the conversation I had previously with the product manager. I shared with him that I had arranged for one of us to participate in a pair testing session with the product manager, and since he felt strongly about the design, participating might be just the opportunity to address some of those frustrations by using a pair testing session with our in-house design 'expert' so we could write up some concrete bug reports about those problems. He became an interested participant, and the session produced both design bugs and a list of ways we could improve upon future pair testing sessions via a retrospective. By the end of the retrospective, both the tester and the product manager were convinced of the value of pair testing.

Helping Others Move Forward
I have learned a lot since deciding I needed to study testing. Taking a step back from my own experience, I realize that the information that I depend on would not have been available if others had not decided to share their experiences with the community. Their willingness to make their own experiences available on Twitter and blogs, to speak at conferences and meetups, and to have online discussions with anyone willing to participate is what allowed me to develop

as much as I have. That self-development is what has given me the courage to make changes within my group. As a result, I feel motivated to return the favor and share my own experiences with the testing community. So far I have presented a poster paper at Pacific Northwest Software Quality Conference (PNSQC) that demonstrated the many testing resources I have found and how they altered my thoughts on testing. Now I am sharing the story of how changing my thinking on testing makes me better able to help the test team at work make changes in how we approach testing. I hope to continue the cycle and encourage other testers to engage in their own studies, make their own changes, and contribute to the testing community.

References

Personal Development
Ministry of Testing. Testing Feeds
Bach, James. To The New Tester
Weekend Testing Americas. About WTA
Markus Gartner. We Are Here to Challenge You (Miagi-Do Blog)
Huesser, Matt. Announcing Software Delivery 24/7

Rethinking Scripted Testing
Holland, Paul. Skills based testing
Austin, Ben. The Software Tester's Easter Egg Hunt
Clokie, Katrina. How to create a visual test coverage model

Pair Testing
Twitter search on Pair Testing. https://twitter.com/search?q=%22pair%20testing%22%20include%3Aretweets&src=typd
McShane, Iain. https://twitter.com/WhyAyeMac/status/400373651553677312
Back, James and Kaner, Cem. www.testingeducation.org/a/pairs.pdf
Moss, Claire. http://www.stickyminds.com/article/exploring-together-shared-understanding-through-paired-exploratory-testing

Helping Others Move Forward
Pacific Northwest Software Quality Conference. www.pnsqc.org