Before We Can Test, We Must Talk: Creating a Culture of Learning Within Quality Teams

Kristina Sitarski
Menlo Innovations
E-mail: kristi.sitarski@gmail.com

Abstract:

Learning is a process you do, not a process that is done to you. Before the heated discussion on what automated functional testing tool to lobby for or the fun game of 'who can cause the exception first,' we must learn how to communicate with one another. In my experience working in a paired quality assurance environment for three years, I have been partnered with several different people. Although my partners have been problem-solving software destructors like myself, they all have had very distinct learning styles. This brings me to a truth sometimes overlooked: to teach is to learn. To help teach not only my partner, but also customers and our Agile shop tour attendees, I look to Neil Fleming's model of different learning styles. Using this as a guideline, as well as some important items ranging from jellybeans to yarn, I find it has improved the communication on concepts such as functional test writing, object modeling, and estimation.

Understanding Mr. Fleming's model is important, but it is imperative to constantly be self aware of the audience you are communicating with and not afraid to create your own teaching style.To deliver quality communication, sometimes on quality itself, we must learn how to effectively teach and learn from one another.

Biography:

Kristina Sitarski has worked at Menlo Innovations as a quality advocate since 2008. Before she became a quality advocate, she worked in quality assurance for Xerox. Currently at Menlo, she brings her experience, as well as a degree in anthropology, to provide an objective, user-centric view of the design and implementation of software tests. She enjoys working and living in the city of Ann Arbor, Michigan.

1. Introduction

Before we can define and deliver quality communication, we must first learn how to effectively teach one another. Effective teaching skills are a concept that can be applied to almost all facets of life, but when given the opportunity to apply it in the workplace (in this case, software testing) it becomes clear there is a dramatic effect on the quality of tests produced as well as the enjoyment level of the testers. I have been given the opportunity to refine and apply effective teaching techniques while working at Menlo Innovations because it is a software company that has developed a culture based on the Agile methodology and human factored design. It is obvious that effective communication is an essential part of this culture, since we practice not only paired programming but also paired testing. It is a common phrase to hear around the factory and among pairs, “make mistakes faster.” To Menlonians, this translates to: making more mistakes gives more opportunities for learning and discussion. The more active discussions we have the faster our knowledge base grows as a team. The “make mistakes” part is easy, but making the discussions meaningful can sometimes be difficult. Like many things in life, the hard part is where we gain the value. These constant discussions are a part of life at Menlo because you are partnered with someone for eight hours of the day. It is the most dynamic collaboration I have ever experienced.

2. A Paradigm Shift

Before my time at Menlo, the whole idea of quality assurance seemed very black and white to me and the words “dynamic collaboration” did not fit into the picture at all. My experience in the QA world to that point had been mostly reading specification documents and following a process. As far as my definition of communication went, a simple one of sending and receiving messages applied nicely. I would receive an e-mail message from my manager, figure out the problem, and send an e-mail back. My manager sat five feet away and yet we rarely spoke. I would rate my performance by the number of e-mails in my inbox. If I was down to twenty messages I felt I was doing a good job. I was also getting good at figuring out problems. In fact, I had figured out a pattern for finding certain types of issues. It never occurred to me to share this information with others. The whole concept of teaching seemed absurd, as I was being rewarded for my work and no one wanted to hear what I had to say. In that environment, speaking outside your realm of work was taken as a challenge to authority or as trying to steal someone’s job. At some point I decided I wanted a serious career or, in actuality, I was afraid I would die of boredom. Through that job, I did discover some important things about myself. I have a knack for finding faults in systems that are not obvious to others. With this new self-realization, I went to search for another job.

With a degree in anthropology, I was not overly confident about my interview with Menlo, a seven-year-old software company, but I would soon learn my father’s inquisitive ways with computers would help me more than I could ever imagine. I was never as much interested in the latest technology as he was, but when one has to step over carcasses of computers in order to get to the laundry room, I suppose one's natural curiosity takes over. I am reminded of the father from the movie, *The Goonies,* except instead of the “Bully Buster glove,” it was an MP3 digital stereo on my belt. As the date of my interview with Menlo drew closer, my nervousness turned to curiosity after receiving the invitation that had the description of “Extreme Interview.” What on earth could this mean? Was I going to be interviewed under some extreme circumstances? When I arrived, there was a group of about 20 people waiting around to be interviewed and, last time I checked, their website had listed only 2 positions open. Things got even weirder when I

was partnered with another person applying for the same job and asked to then, “help you partner get the job.” Are these people crazy? Have I accidentally walked into a speed dating event?

We sat down across from a current Menlo employee who was there to observe our “kindergarten skills” as we tried to perform simple problem solving exercises together. What seemed terrifying soon became refreshing. I was in a group of people that actually cared not just about having the problem solved, but about the *way* in which it was solved. Even though I was working with complete strangers, I remember feeling so creative and inspired that I forgot I was in an interview and eventually started thinking, “this is fun.” I am fortunate to say I still get to have fun and get paid. The uniqueness of that interview carried on into the everyday work environment, as I now have a partner who I work with every day. It’s just like all those famous cop shows, except instead of killers we are chasing down database errors. When I started my new QA job, I would use the application, find a bug, and look to my partner for a congratulations. Instead of congratulatory, the look on their face was perplexed and almost slightly annoyed. They would ask, “why did you click there? Why did you do that?”

I didn't know how to respond. I felt like I had suddenly forgotten the English language. Why couldn't I verbalize my train of thought? Could I just send them an e-mail explanation later? I quickly realized that my old model of communication was not going to work.

I tried my best to talk out my thought process as it was occurring, but it seemed like that only resulted in sentence fragments and rapid bursts of excitement. Finally, I found myself grabbing scraps of paper and drawing pictures. Most of the time these pictures made little sense to anyone but me, but the important part was that, somehow, if I had a squiggly line to point to, it would help the words come out of my mouth. It wasn't the picture, but the conversation it incited between me and my partner. As I had more practice at this, it became very important to make sure the other person understood why I was doing something. Sharing thoughts in a way that was effective and organized allowed my partner to contribute to the problem at hand in a more meaningful way. It was collaboration and it was fun. The next week, I was assigned to a new partner. I was now the senior in the pairing and it was no longer adequate to have the skill of being able to explain my thoughts regarding a specific test setup. I now had to explain high-level architecture and present testing plans to clients. Things were not going smoothly. We would discuss a plan of action, but then my partner would execute something completely different. As it turns out, my partner was confused so much by my meaningless squiggles and lines that he couldn't focus on what was being said in the conversation. He would agree to the test plan, but his plan was something different than what I thought we had just discussed. This was when I realized, just like with my first partner, we needed to figure out how to talk and start truly communicating.

3. Learning to Learn

I began researching Neil Fleming’s model of different learning styles. In this model, Fleming proposed that one could fall into three different categories of learning: visual, auditory or kinesthetic.

> Fleming claimed that visual learners have a preference for seeing or thinking in pictures, using visual aids such as overhead slides, diagrams, handouts. Auditory learners best learn through listening, lectures, discussions, or tapes. Tactile/kinesthetic learners prefer to learn via experience—moving, touching, and doing, active exploration of the world; science projects; experiments. (Learning Styles Explained 2008)

Fleming had even developed a questionnaire that would determine your style of learning. My partner and I were very excited to discover what category we fell into and we both took the survey. Fifteen minutes later, the results were in; I ranked highest in the kinesthetic category and lowest in the auditory. My partner ranked highest in the auditory and lowest in the kinesthetic. We were delighted by all this data and quickly went to the next page of the questionnaire in hopes that this is where we would find all the answers to our problems. Of course the next page read, "there are individualized learning profiles (pages of advice) available for purchase."

What followed next was a discussion of what all these categories even mean and if we even care. What we took from this was the ability to be more self-aware while teaching. It was important to develop a way of communication in a style that was more likely to be understood rather than the style in which the teacher understands things. It is a very simple concept. Understanding it is the key to self-awareness. By researching the different learning styles I became more aware of when I was about to go off on a tangent; talking in metaphors and opening 500 different tables in the database. We exposed ourselves to different ways of teaching and decided that just one method of teaching would not work.

Menlo, as an organization, was more advanced in this regard than our small quality advocacy team. I started noticing in classes and in the tours that were offered to the public and new clients, how presenters geared their teaching towards all of the different styles of learning. A great example of this was an exercise that was developed to help a new team understand the definition of object oriented design and what an object model is. It was especially helpful during a time when Menlo was transitioning a project back to a development environment located at the client site. The clients had just hired a new team and were now ready to take over two years of code written at Menlo. Many of their programmers were getting frustrated at not understanding why the code was sometimes written in a way that seemed like more work in order to produce a fairly simple function. The client sponsors, who are non-technical people, were having a hard time understanding their team's frustrations. To help solve this problem, we had a meeting involving both teams representing roles such as: developers, QA, project sponsors, and solution architects.

Step One: we divided into mixed groups of teams across roles.
Step Two: someone read aloud the brief definition of an object model from a reference:

> An object-oriented program will usually contain different types of objects, each type corresponding to a particular kind of complex data to be managed or perhaps to a real-world object or concept such as a bank account, a hockey player, or a bulldozer. A program might well contain multiple copies of each type of object, one for each of the real-world objects the program is dealing with. For instance, there could be one bank account object for each real-world account at a particular bank. Each copy of the bank account object would be alike in the methods it offers for manipulating or reading its data, but the data inside each object would differ reflecting the different history of each account. Objects can be thought of as wrapping their data within a set of functions designed to ensure that the data are used appropriately, and to assist in that use. (Kay 2003)

Step Three: making sense of the artifacts in front of us on the table, which included: stacks of different colored note cards, rulers, markers, and three different types of graphing paper.

The goal was to build a mock graphing calculator using object oriented design. Soon the room was full of discussion. What are the objects we need? What are the functions of those objects? How do they relate? Soon we had project managers writing Class-Responsibility-Collaboration (CRC) cards saying things like, "but why isn't our data set object in charge of drawing?" It is important to note that not only did we decide

on what our objects would be, but we acted as if we were the graphing calculator ourselves. We were physically drawing the points and lines that were the output from our math equation object we had created. This triggered ideas that would have been lost if we simply had a real graphing calculator do the work, or worse, just conceptually thinking about what the calculator would do. It also gave an outlet for ideas such as, “What if our graphing calculator was voice activated? Can our lines draw in different colors?”

To wrap up, each group presented on what they had accomplished. In a matter of two hours we had used every learning style Fleming could have imagined and created a common language that was shared between technical and non-technical people. The team learned a style of collaboration that made for quality communication. The team also took away a better concept and appreciation for object oriented development and design.

4. Conclusion

Building and testing great software is hard work; talking should be easy. It is always difficult to start a cultural shift in thinking among teams, especially because it must come from the team members themselves. A manager cannot enforce a cultural process of good communication. The team must build the habit of talking aloud and with one another. They must support each other and know when to suggest moving the problem -solving away from the computer and to a white board. It is imperative to constantly be self-aware of what you are doing and why you are doing it. You should always be able to teach your partner your thought process and talk through why your testing steps are organized in the way they are. It is easy to suggest, “oh it’s just random” or “I just have a hunch it might break there,” but it is vital to talk through these steps more thoroughly, while keeping a person’s personal learning style in mind and constantly sharing information.

Constant communication not only fosters learning but creates a safe environment where people feel it is okay to share these small details and where questions can be asked that would normally be overlooked or deemed not important. In actuality, the smallest thought could be something that turns the light on or fills a missing link for someone else. Before you realize it, you are learning something completely new and a problem is on the right track for being solved rather than still being worked on alone for hours. The values that make a great teacher are similar to the values of a quality team: clear, written-out objectives, looking at issues in a variety of ways, having a strong sense of caring for the end user and, above all, a passion for learning.

References

1. "*What are learning styles?,*" LdPride. (n.d.)., accessed October 17, 2008, http://www.ldpride.net/learningstyles.MI.htm#Learning%20Styles%20Explained.

2. "Dr. Alan Kay on the Meaning of "Object-Oriented Programming"," Kay, Allen PhD, 2003. accessed 2010-02-11, http://userpage.fu-berlin.de/~ram/pub/pub_jf47ht81Ht/doc_kay_oop_en
.