Mob Programming – My first team

ROBERT "JASON" KERNEY, Hunter Industries

I joined a team that previously discovered a new way to work, that was called Mob Programming. As the newest member, I expected a total team approach to everything and a heavy focus on agility. However, I did not expect the experience to be as transformative to me as it was.

1. INTRODUCTION

I am an individual and a part of a team. That sounds pretty normal, but isn't. In Charles Duhigg's book, *The Power of Habit*, he talks about how American football teams are not so much a team as a group of people playing together. This is exactly how I feel now that I have been doing Mob Programming. I feel that most programming teams are not so much teams as they are people who work together. I *am* part of a team.

Before explaining what I mean by "I am part of a team," I feel that I need to give some context. I work at Hunter Industries, one of the largest makers and distributors of outdoor lighting and irrigation products. The team I work with focuses on back-end business functions.

The Hunter mob formed out of necessity. They had been practicing every morning using a Coding Dojo format for an hour. The team worked the way traditional teams worked, but they were also experimenting with pair programming. Then came a problem that no single person had the knowledge or skill to tackle. The team swarmed. They congregated in a meeting room to figure out how to approach something this big. After a few minutes, the team decided to pull the code up on the screen. The habits they had formed doing Coding Dojos guided them as they started working through the problem.

Then the time for the meeting ran out. The team scrambled to find a new meeting room. After a week, the team had a retrospective and found it to be a very fun and productive way to work. They decided to try it for everything; thus Mob Programming was born.

2. Our Mob

2.1 Overview

I work with some amazing people and our success is related to those people and their willingness to explore and learn. These people set the groundwork I encountered upon joining the mob. This groundwork formed a structure that I did not have to discover; it was already there for me to benefit from.

Mob programming is all the people working together on the same thing, on the same computer. What this means for us is that we have 5 programmers sitting in a space with our manager. We use projectors and special projector paint to display our work. We each have our own chairs that are configured ergonomically. The projectors are low so that we do not have to look up. Everything about our configuration is designed to make us comfortable and increase sustainability.

We operate by a set of three rules and a bunch of guidelines. The first rule is everything we do we do with kindness. The second rule is that we consider everyone's opinion. The third rule is that we show respect to each other and others. Simple rules, but profound. Think about how different your day would be if everyone you interacted with acted with kindness, listened with consideration, and showed you respect? I honestly believe it is the focus we have on these rules that guides us to success.

2.2 Rules

When every interaction you have is guided by kindness, consideration, and respect you gain a security and a sense of freedom that just normally would not be there. This security helps with the learning and the willingness to discover. There are also secondary effects. Imagine that you are having a technology discussion. Let's say, "Tabs vs. spaces." Your coworker strongly disagrees with you. We all have been there. How do you resolve it? I have literally witnessed this very "discussion" cripple a team as they fight over what format is committed to source control. When approaching this with kindness, it prevents name calling and derogatory remarks. Taken with consideration, we must realize there are reasons for the other person's point of view. When approaching it with respect, you have to assume that the other person is trying to genuinely do their best for the team. Conflicts like this dissolve faster, because we can kindly argue our point, we consider that point of

Author's address: 1940 Diamond Street, San Marcos, CA 92078; email: Jason@teamkerney.com

view, and have enough respect to allow others to win. More importantly, we know that our opinion is given respect in turn. It is hard to argue by personality when everyone there expects kindness, consideration and respect. They will in fact kindly and respectfully tell you how you are acting with consideration for your point of view.

2.3 Guidelines

2.3.1 The timer

The rules are ever-present and immutable. The guidelines, however, are constantly deviating and being explored. The most visible guideline we have is our timer and our rotations. Our general mode of operation is that we rotate who is at the keyboard every 15 minutes. This rotation is randomized from day to day.

The process of discovering how to do this happened before I joined. It started as a kitchen timer that would beep every 15 minutes, signaling a rotation. I am told that the rotation did not always occur when the timer went off, as the person typing wanted to finish what they were working on.

It was discovered by a team member, by chance, that the audible timer was distracting and irritating to other people on the same floor who did not know what it was or benefit from its use. Quickly, the team generated a timer application. The application had to get your attention without making a noise. They chose to make the application block out the whole screen and show who was next. This both removed the noise, and encouraged immediate switching of who was at the keyboard.

This does not mean that we always rotate when the timer is up, but it is a decision of the group now. We also do not always use 15-minute rotations. We change the rotation times for special circumstances, such as for visitors or learning sessions.

2.3.2 Driver and Navigators

The next big guideline we have is the “Driver” and “Navigators” roles. The people not at the keyboard are the Navigators. It is their responsibility to discuss and formulate what should be coded. It is the responsibility of the person who is at the keyboard, the Driver, to translate the ideas and guidance of the Navigators into code. In theory, all production code has to go from at least one Navigator to the Driver then to the code.

This applies to emails, code, check-in comments and even google searches. The reason I said ideally is that it sometimes bogs down the team. Sometimes we all are doing separate google searches or someone is calling a businessperson to answer a question. Other times we are changing it up to give a member of the team a chance to express themself. These are the exceptions though.

2.3.3 Study Sessions

We have study sessions every morning for an hour and every Friday end of day for two hours. This is specifically for sharpening our metaphoric saw. During these sessions we focus on refining our good habits or exploring new technology. What is interesting is that these function differently from regular work. The first thing we do to signify a learning session is that we shorten the timer to five-minute rotations, meaning a new person at the keyboard every five minutes. Then, depending on the subject and familiarity we experiment with the roles of Driver and Navigators.

2.3.4 Automation

The next guideline is that we automate everything. If we find we do something twice and it is not instantaneous, we automate it. It is amazing to watch the difference between wasting one minute of your time versus wasting one minute of five people's time. We strive to make things as fluid as possible so that we do not have to break our flow.

This has a number of benefits. The first is that we have an environment that is easy to maintain. The tricky parts of maintaining it have been automated, or as close to true automation as you can get. It also means that there is less error on boring tasks. Even five people cannot focus all the time on something that is tedious. Lastly, it means we have a greater confidence in what we do. Since there is less chance of making mistakes on the tasks most prone to mistakes, we fear doing them less.

2.3.5 Retrospect

The last guideline I would like to talk about is our retrospectives. We tend to do immediate and short retrospectives. As soon as someone on the team notices something, good or bad, we all roundtable it. It happens right there when it is fresh and we have all the facts. We also tend to focus on good things for retrospectives. When we see something that went surprisingly well, we want to capture it and build upon it. It doesn't mean that if something needs to be fixed that we are shy about doing it. We recently realized that the projectors we had were constantly going dim, and it made the text hard to read. We had several experiments and retrospectives related to those experiments until we found a solution.

2.3.6 These are Guidelines

There is an important thing to illustrate here. The guidelines are there for a reason. Yes, they are allowed and expected to be in flux; however we do not ignore them. We go through a lot of effort to ensure what we are doing is the best we can do. We look at our guidelines and evaluate them regularly. If one has started to fade we ask why and ensure it is intentional. If not, we adjust our behaviors to bring that guideline back into the light.

3. Applying for the Mob

I did not join the Mob by accident. I knew Woody Zuill [Zuill], Programming Manager for the mob, for about 10 years before joining. I worked with him at a past company and consider him a mentor. I knew about Mob Programming from discussions with Woody and by having attended a couple of talks he led on the subject. I had even turned down an offer to join earlier because of timing. So when I contacted him about the job, I wasn't even sure it was available.

The interview process was interesting. I knew that Woody believed more in hands-on testing for interviews, but I also knew that he had a high regard for my abilities, so I was not sure what to expect. My first experience with the team was the team deciding how to interview me. I was added to the timer, and the interval was set to 7 minutes. Then we worked on the Tic-Tac-Toe kata for a half an hour.

After coding with the team, they were curious about how I manage my continuous education, and they asked me to show them something they can add to their learning sessions.

I met with a couple of executives within the department that informed me of the substantial financial benefit Hunter has received since the team started Mob Programming. I was also told I should not join if I am not sold on programming this way. That was the extent of the interview, and shortly after, I received my offer.

4. Joining the Mob

Onboarding is a dirty term. It is the sludge a company makes you wade through just to be able to do the work you were hired to do. I have seen organizations where it takes a month before you write your first line of code. The Hunter Mob was very different. There was absolutely zero technical onboarding. After doing the things required by HR, I was taken to the mobbing area and added to the rotation. There was no explanation of the current task, and only the barest description of what the project was. I was working in production code.

I quickly discovered that the project was a web-based application. I was not a web programmer. In my 18 years' experience I had done a total of five hours of web programming seven years before. I should have been fighting to stay afloat, but I wasn't.

The project has a monthly meeting, where the executives of the four departments responsible gather to hear a status. They want the opinion of each team about what has been accomplished and what obstacles that might need executive authority to be able to remove. This meeting was that Friday, and I was able to lead it.

5. Learning as A Team

Mob Programming speeds learning in three different ways. We are working on production code 34 hours a week. During that time you are on the keyboard for 15 minutes out of every hour and 15 minutes. That means you are the Driver 82 minutes every day on production code. That translates to 6 hours and 48 minutes a week. During that time you are being guided and examined by experts in what you are working on. When you are not driving the rest of the 33 hours, in that week, you are discussing the product, the design and the implementation. Imagine how well at something you would be if you spent 7 hours a week under direct guidance of a tutor, and 33 hours a week in discussion with that tutor?

So what happens if no one is the expert at what you are doing? If there is no direct tutor to learn from, then what? Here Mob Programming speeds learning by allowing you to lean on each other. No one has to be brilliant; all they have to do is be willing to provide input when they have it. Then a strange thing happens. Because of Transactive Memory (http://www.danwegner.net/tm.htm) the group is able to draw on a deeper body of knowledge than the simple sum of the individuals. They can then use that knowledge as jumping off points for research and study.

Lastly, we are not afraid to experiment. This is beyond our learning sessions. We are willing to burn a week on an idea that sounds as if it can be helpful to our process or a project. Some of our most amazing things have happened because someone said, "I have an idea. I only wish I had it a couple of months ago, because it would have been easy."

For example, we needed a way to test that security configuration for an application was done correctly. To further complicate things, the configuration was still changing. So we needed a way to take a written specification, convert that to a configuration, and then use that to test the application. We did not know if it was possible. We spent a week exploring and tinkering. The whole time we were thinking we might throw the entire week away. At the end of the week we knew we had a direction and by the end of two weeks we had it implemented and tested. I have never been given that kind of freedom before, and because of it we accomplished something amazing.

6. It was not magic or instant

I did not become part of the mob instantly; it took work and willingness to learn. The first problem I faced was exhaustion. I enjoyed every moment right out of the gate, but by the end of the day I had no energy. I wanted nothing more than sleep. This meant less time for personal projects and more importantly my wife and kids. I thought I would eventually get my stride and everything would ease up. After all, mobbing did not seem to affect my teammates the same way.

I could not keep up with them. I worried every time I left to get coffee or use the restroom that I would fall behind. I made these reprieves as short as possible, and I never fell behind. I did not know how long I could keep it up though without burning myself out.

A little after a month, I learned my most important lesson. I looked around and noticed that my teammates would occasionally leave, just to walk away. Sometimes they checked personal email on their phones. They were actively taking breaks. So I stood up, and leisurely walked to the break room. I talked to a co-worker from a different team. The whole time I was slightly panicked that when I got back I would be behind.

When I returned, the strange thing was I was not behind. The way mobbing accelerates learning made the seven minutes I was gone condense into less than a minute of catch up time. That night I went home and played board games with my kids. I suddenly felt secure in what I was doing.

Sometime after that came another powerful lesson. I am often rude. I do not mean to be but I am excitable, passionate, and occasionally unaware. This lesson came to me as a sudden shock one day when I was very excited about something being said. Right in the middle of a sentence I hijacked the conversation, speaking right over my coworker. As it happened I suddenly became aware that my coworker was uncomfortable and a little annoyed. He said nothing; I just noticed it on his face. I stopped and I apologized. I surrendered control of the conversation. Afterwards, I started trying to be more cognizant of my actions. After a couple of days noticing how often I did something like that I slowly got better. I asked one of my coworkers to help me see it when I was being rude. I am continuously striving to be a better listener, and better aware of other people.

7. Not Advanced Pair Programming

When I joined the Mob, I thought I knew what Mob Programming was. I expected it to be exactly like Pair Programming only with more people. I was wrong. The problem with comparing it to Pair Programming with more people is that Pair Programming is all about the dynamic of two individuals programming on the same thing at the same time at the same computer. Mob Programming changes that dynamic. It has to, if for no other reason than you have to accommodate communication of a group, versus a straightforward conversation between two individuals.

Mob Programming shares all the aspects of Pair Programming: continuous code reviews, the ability to lengthen focus, the attention to good details, and the ability for someone else to catch errors are just a few. Each of these practices is amplified by each person involved. In this way, Mob Programming is an advanced form of Pair Programming.

There are additional benefits though. The mob is not crippled if you have to go to the bathroom, go on vacation or even leave early to pick up your kids. The fact that there is redundant knowledge allows you to leave the mob and know the job is still going to get done.

The number of people in Mob Programming increases the gains from systems like Transactive Memory. Transactive Memory is a process where multiple people who share an experience store different pieces of that experience to allow for shared recall, and reduction of duplicated effort. This process was heavily studied in couples, but researchers believe it might also apply to teams and friends. I have witnessed the mob remembering something in a manner very similar as what the research described.

The last gain I want to talk about is a process I liken to wolves howling. You see when a pack of wolves howl, no two wolves howl at the same frequency. In coding, there are a lot of things to think about, architecture, design, the problem at hand, coding standards, testing, deployment, business impact and security to name a few. I have found that our mob treats each of these like a wolves' howling frequency. We each take one or a few and pay attention to those. If someone else appears to be covering it, we choose something else. None of this is an active decision, it just happens. Pair Programming, because it only has two individuals, means that there is a lot more burden on the individuals to track each of these things. I have never seen a group adapt to each other's strengths and attention in real time before. It is relieving and freeing to be a part of.

8. Different Points of View

Just because we act with kindness, consideration, and respect does not mean that we always agree, nor does it mean we want to always agree. One of the questions that comes up about Mob Programming is, "What do you do about strong personalities?"

Well, I am a strong personality. I am not the only strong personality on the team. This has led to passionate, yet respectful, conversations. Our team will make suggestions during these conversations. Some of the suggestions are about the topic; others are gentle reminders to listen to what the other people have to say.

Conflict though is not the only outcome from different points of view. We benefit from these conversations. We are not afraid to try, and when we are done, we try again. When no obvious answer presents itself we try everything. We take the better of the options, which often ends up being something else that we discover while trying these options. Also, sometimes, we discover that the discussion did not matter once a solution is implemented.

This allows us to explore and learn from each other in a whole new way. We can randomly choose an option and go with it. We can explore ideas that any of us individually normally would not try. Our different points of view and experiences are important to us.

9. Team Work

I have never worked in an environment before where everything is collaborative and with no barriers to communication. There are studies that suggest that simply increasing the distance between two people on the same floor reduces the amount of communication between those individuals. Here, I work feet from other people on my team, and there is not as much as a table between us.

When you combine that with continuous rotations, Transactive Memory, communal study sessions, subtle modification of attention to better cover the whole code … this is what I mean when I say I am part of a team. When people say the word teamwork, we usually mean work divided up amongst the team. When *we* talk about teamwork, we mean we work as a team. At this point, I can't picture getting that level of team work from a non-mob, which would make it challenging for me to ever move again into a non-mob environment.

10. Support Structures

In the team I am on, we have really good support structures. We help each other when we need it. The team openly recognizes that problems will arise. We also understand that each person has his own needs and goals. We strive to support those goals.

One team member builds games in his spare time. We believe that the knowledge he gains by pursuing this is beneficial to him. When a game development conference comes up, he goes.

Another team-member wanted to work on his typing. So what that meant was that when he was at the keyboard his typing was slower and had more mistakes. We understood that he is intelligent and that this inefficiency will pass. So on his turn we navigated slower, and sometimes dictated what needed to be typed so that he could focus on only improving his typing.

I did not know web programming when I first joined the mob. The team took time to guide and explain, and also guide me to resources when my questions prompted it. They slowed down specifically to ensure that I learned and understood what I was doing. This was done because I wanted it, not because the team decided it.

Lastly, two team members, including myself, decided to switch keyboard layouts. This affected the whole team. We had to install new keyboard drivers on all servers and virtual machines that the team deals with. Also we had to deal with the fact that the team now used 2 different layouts. We were able to figure out a way to make this easier for the team without coalescing on a single keyboard layout.

11. The Perfect Mob Programmer

Inevitably during a discussion on Mob Programming someone says, "Mob Programming is not for everyone." The corollary question is, "Who is Mob Programming for?" That is a hard question to answer. There simply is not enough data. My mob is generally really careful about "selling" Mob Programming. We are not salesmen because we do not know if mob programming would be good in any given situation.

We are an awesome team that practices Mob Programming. However the cause and effect of that statement are hard to sort out. Is it because we are an awesome team that we can Mob Program? Or is it because of Mob Programming that we are an awesome team?

What I can tell you is that it takes trust. If there is some part of your job/career you are not good at, it is obvious to everyone. You cannot withstand that kind of pressure without trust in your team, company, and self.

You have to trust that they will see and recognize not only your growth but also the other areas where you shine. You have to have trust in the rules; that you will be treated with kindness, consideration, and respect. You have to trust that the treatment is genuine and not faked. You have to trust that everyone is acting toward the benefit of the group, not just themselves.

You have to have a lot of trust. If you do not have that trust, I can believe that you would easily feel uncomfortable. Without this trust, it is also likely that the scrutiny of the situation will make you feel exposed and at risk.

I do not know what makes a perfect Mob Programmer, but I believe without trust from you and your team, your team will not be very successful at Mob Programming.

12. Personal Note

Lastly, there was an unexpected benefit I have gained from working with my team in this fashion. As I focused on listening to people more at work, I started listening more to people outside of work. I became more attentive to when I interrupt people or hijack a conversation.

A few months ago, my wife told me that she noticed how I had become a better listener. She was worried that telling me would be offensive; however it elated me. I had not noticed how things I was practicing at work were bleeding over into my home life. Suddenly I had confirmation that everything I was doing was working and positively affecting me. Mob Programming made me a better person.

So I am a member of this amazing team, but I am me. I have my own goals, my own life, and I bring my work home with me. I also bring my home to work with me. This experience really made me aware of this. If and when I leave Hunter, I will crave this interaction. I cannot picture me being *on* a team without wanting to be *with* a team. This is the difference between now and every other project and company where I have worked. I have always been on a team, but never part of one. Now I am part of one.

13. Acknowledgements

I want to thank Woody Zuill and Llewellyn Falco for being mentors when I needed them. I want to thank the "Hunter mob" as a whole for allowing me to join them. Thank you Hunter Industries for allowing this magic to happen. A special thanks to Carol Kerney and Catherine Kerney for reading and editing this document. Thank you Rebecca Wirfs-Brock for asking me to write this report, shepherding me in its writing, and keeping me on it. Most of all I want to thank her for identifying the gems within my writing and helping me in polishing them.

REFERENCES

[Zuill] http://www.agilealliance.org/files/6214/0509/9357/ExperienceReport.2014.Zuill.pdf