

Using Competition to Build a Stronger Team

Darin Cummins
ProQuest Powersports
darin.cummins@pbs.proquest.com

Abstract

In 2001 we started a new project at our company. Undermanned, short on time, and under the gun to succeed, we knew that we needed a process that would help us stay on track. Unfortunately, once the project got under way, we received a lot of negative feedback from the developers. The process took away from what developers want to do: code.

This paper describes how we used competition as a tool to create a more cohesive team that worked better with management and the process.

1. Who we are

ProQuest is a large company divided into two primary divisions: Information and Learning, and Business Solutions. ProQuest Powersports is a division of the Business Solutions side of the company where we develop software solutions for the power sports industry. We are located in Salt Lake City, Utah.

We have two primary products for the power sports industry. The first is an electronics parts catalog designed to replace the old microfiche technology. It has the ability to locate parts, zoom in on images and “exploded” diagrams of motorcycles, etc.

Our second and larger product, is a complete dealership management system. It includes 9 modules, purchasable in various combinations to help dealerships of all sizes run their business. The current version of the software is the third rewrite in 20 years and is a J2EE based product running on both Linux and Windows.

Our management team is comprised mostly of developers who approach development as a team effort where we all work together. Most of the managers, including myself, spend as much time as possible designing and coding along with the other team members.

2. Starting a new project

In 2001, we decided that we needed to rewrite the product to bring it into the 21st century. Two of us were tasked with putting together the project, investigating technologies, etc. We enlisted the help of other teams and spent about 5 months determining the best course of action based on an initial set of requirements. After much deliberation, and many prototypes, we decided that J2EE

provided the greatest set of tools and technology for our goals.

After determining the technologies we wanted to use, we needed to put together a team. We didn't have the Java expertise in house but we did have a lot of long-term developers with extensive domain knowledge. We decided that the best solution would be to build the team from a combination of existing domain experts and new technology experts, hoping that the domain experts would pick up the technologies quickly and that the new guys would pick up on domain knowledge just as quickly.

3. Implementing Process

Prior to this project, our experience with process was very hit-and-miss. We knew that we needed process and we could see the problems that we were having, but we didn't know where to start in setting up a good process. We decided that this new project was a perfect opportunity to put something in place that we could build upon with future projects.

The manager of the legacy team, who was one of the people enlisted to help us put the project together, looked at some options in the process area. After some investigation, he thought that one of the processes described by Scott Ambler[1] would be a good starting place. We made some modification to what Scott described, and since the manager's name was Reece Newman, we decided to call the process the “Rambler” method.

4. Troubles in Paradise

Once we started the project and implemented the process, we began getting feedback from the developers. Mostly the feedback was negative. The process that we were trying to implement took too much time from the developers actual work. They were complaining about having to fill out worksheets, etc. But mostly they complained about code reviews.

In the past, we had done limited amounts of code review and, while it was time consuming, it did seem to have some benefit. Now, though, we were realizing that to really review all the code that we wanted was taking far more time than the actual coding. We tried to make the process more efficient by conducting weekly reviews at lunch time with management supplying the food. Unfortunately, all we gained was having developers look at the code to be

reviewed for about 5-10 minutes before the meeting. There just wasn't enough time to get reviews in.

What we came to realize was that we didn't understand enough about process and tactics to know where we needed to make changes and where we needed to go. When developers started to complain, we knew that we needed a process but we couldn't even effectively defend why we needed a process. Needless to say, morale was low and it didn't seem like we had an easy solution to the problem.

5. Game Design

In September of 2002, a friend, Jeff Patton, invited me to the Salt Lake Agile Group, which is a round table that meets once a month to discuss software development ideas. That was my first introduction to agile methodologies and started a whole new phase of development for our team.

I began looking more thoroughly into the principals and ideas behind Agile development and began talking to the thought leaders of our project about implementing some of the ideas. Although skeptical about some of the ideas, we all knew that there was a problem and we were willing to try some new things to get developers back on track.

During one of the round table meetings, one of the participants, Kay Johanson, mentioned an article in the Harvard Business Review by Phil Orbanes, called "Everything I know about business I learned from Monopoly" [2] that discussed game design principals and how they can apply to building and running a business. I was intrigued and purchased the document on line. In the article, Phil talks about 6 primary guidelines to follow when designing a game:

1. Make the rules simple and unambiguous.
2. Don't frustrate the casual player.
3. Establish a rhythm.
4. Focus on what's happening OFF the board.
5. Give them chances to come from behind.
6. Provide outlets for latent talents.

Phil discusses ways of applying these rules to management teams and creating a work environment that had the same kind of enjoyment that people get from games.

After reading the article, I thought that I had found a possible solution to our problem. I began to wonder what would happen if we could make development a game. Players could earn points for what they were required to do and then we could have rewards for the points at the end of a cycle. After pondering the idea for a little while, I approached the other team leaders and brought up the idea. We hammered at the idea for a couple of weeks, revised some of the initial thoughts, and eventually came up with the game.

This may not have been how Phil had intended his article to be utilized, but it did work for us!

6. The Development Game

We created a "Pay Chart" to encourage developers to do the things that we needed them to do as part of the process.

Description	Amount
Developing	
Creating Unit Tests / Test Plans	\$20
Creating Supplemental Diagrams	\$20
Creating Engineering Document	\$50
Code Review	
Review code for someone else	\$30
Receive an "Excellent" average rating (2)	\$50
Receive a "Acceptable" average rating (1)	\$20
Receive a "Unacceptable" average rating	-\$20
Don't have code reviewed	-\$50
Implementing suggestions from Code Review	\$30
Code Reuse**	
Implement code someone else has written	\$20
Writing code that can be reused by others	\$20
Utilities and Tools	
Writing a tool that improves development (coding, process, etc.)*	\$10- \$200
Cycles	
Completing Cycle on Time	\$30
All expected Functionality Complete	\$20
Closing a Functional Task Completely	\$20
Off scheduled task without team or management approval	-\$20
Miscellaneous	
Overtime	\$10/hr
Help someone to learn a new part of the technologies *	\$10-\$50
Time Entered on Friday before being reminded	\$10
Build	
Errors in build (build or deployment failed)	-\$50
Errors in Tests (build passed but tests didn't)	-\$30
Warnings in build (deprecations, etc.)	-\$20
Quality Assurance	
Rollover	-\$20

* Marked items require team consensus.

** Code reuse is meant primarily for unplanned reuse. Otherwise, the framework and system experts, who are by definition creating code for everyone to use, would be richer than the others.

We created fake money by using photographs of our Vice President and putting it on photo-copies of various denominations of money. (Did you know that HP printer drivers will recognize that you are printing money and print a warning on the page?)

At the end of each cycle, we had a payday. Everyone had to fill in a small chart and give it to their manager. The idea of short cycles, by the way, came from discussions at the Agile Round table.

We also planned some other activities where people could make up any shortages they might feel they have. These activities included things like "count the M&M's", trivia games about people in the office, etc.

When we first started developing the game, we weren't sure how to approach what the money would be good for. We contemplated creating a "company store" where the fake money could be used to purchase things like hats, t-shirts, etc. But we couldn't think of anything that really

sounded motivating. Then someone had the idea of an auction.

We talked upper-management into allowing us to spend \$1400 for items to auction. Compared to what other teams were asking for as incentives, this seemed a small amount.

The management team then spent time purchasing items ranging from \$5 to \$200. Obviously we could not have a lot of “big ticket” items, but we did get a couple of nice things (camp stove, tool set, etc.) We made sure that we had enough items so that everyone on the team would go home with something, even if it was a humming-bird feeder.

7. Playing the Game

We started the first iteration of the game with a little trepidation. Some developers were really gung-ho about the whole thing and others mostly stayed quiet until they knew how things were going to work out. But even those developers weren't willing to be left out, so they participated in a much quieter manner.

After the first 3-week iteration, when everyone got paid, there was some quiet discussion among the developers, asking “how much did you make”, etc. Management still wasn't sure at this point whether it was going to work out or not, but some of the complaining seemed to have subsided.

By the second iteration, we had some of the items purchased and placed in a location where everyone could go see them. As excitement built over the items, so did the comparisons of amounts earned, etc.

We had many small competitions during the game but there were a few that were very memorable. For instance, we had a pop drinking contest where the winner had to run to the bathroom the instant he was declared a winner. In fact, I think the bathroom floor is still orange! Another was a trivia game we played where we had to guess things like “How many MP3 files does Mike have on his machine?”

At one point we had one guy who ended up being out of the office for a while. To help him catch up, we increased his pay scale for a short time. Otherwise, he would have just stopped playing the game since he was behind.

During the course of the game, a noticeable thing occurred: people began to collaborate on task in ways that would allow each to earn more money. People helped each other to write tests, to create code that was reusable, etc.

By the end of the game, people were really ribbing and harassing each other. Some had openly admitted that they were going to bid on specific items and others seemed to take that as a challenge to beat them out of it! At this point, everyone was involved.

One of the things we realized during the first game, is that people like money. The more money they have, the more they like it. So we decided to increase the pay amounts by 10 times. So an activity earning \$10 became worth \$100. This was well received.

8. The Auction

When the time finally came for the auction, we talked one of our fast-talking salesmen into being our auctioneer.

We had the auction on a Friday afternoon, and supplied lunch as part of the festivities. We allotted about 2 hours and all gathered around a small conference area we had set up in the middle of our work area.

Auction items were chosen at random. Each item was given a number and the numbers were drawn out of a hat as the auction proceeded.

The competition during the auction was intense. People were harassing and teasing each other, there was yelling across the room, people were laughing so hard they couldn't talk, and our auctioneer was well on his way to becoming a stand up comic!

What was really unexpected was the back-of-the-room negotiating that began to take place as the auction came closer to the end. People knew they didn't have enough money to out-bid another person so they began sharing money. In some cases, they were buying already purchased items from each other to increase their holdings!

All in all, it was a great time. Everyone went home with something and everyone wanted to do it again soon.

9. Results of the First Game

The effects of the first game were very pleasing. First and foremost, the negativity seemed to have abated. While there was still discussions about what wasn't working in the process, developers were being more constructive as to how we could solve problems, rather than just complaining about management not being very smart.

During the running of this iteration of the game, I had attended a few more meetings of the Agile Round Table and had begun to invest more time in exploring what other teams were doing. One of the things that was intriguing, was the concept of pair programming. We had a few discussions during team meetings and the general consensus was that pairing couldn't be as productive as others were claiming.

As we continued through the game, though, we actually ended up with some developers trying pair programming to help increase their productivity. The funny part is that the original skeptics are the biggest advocates of pairing today!

10. Observations from the First Game

There were a few things that we observed during the first game that we applied to later iterations of the game.

Higher amounts of money are more motivating. People like to earn \$1000 for something instead of \$100. Even though the playing field is even when everyone gets paid the same, there is something about having large stacks of money in your hands on payday.

We originally talked about just giving points for the

various items, but having the fake cash was a huge success. The paper money, whether real or not, provided a visible and tactile substance to the development efforts.

Rather than having one person purchase all the gifts, we had a number of people shop. This gave a greater variety in the items and provided some fun for the managers. Besides, having one person responsible for all the items is a lot to ask since the shopping has to be done on personal time.

We had one person handle all of the money. That allowed our “working managers” to participate in the auction also.

We included gag items (a brass pig, a humming-bird feeder, etc.) as well as the more desired items. It was fun because people would bid really low on those items (\$1) where the big ticket items were going for larger amounts (\$8000).

The small activities that we had during the game were important. They provided an opportunity for those who may be behind in earning money a chance to catch up.

11. The Second Time Around

The second iteration of the game was played almost a year after the first. Developers were helping with the process, making suggestions and for the most part everyone was working well together. While things were going well, there were still some things that we wanted to implement into the process and things seemed to be getting a little too routine.

This time we structured the game to focus more on teamwork, rewarding teamwork skills like pairing, communications, etc. more than other skills. We had also learned our lesson with code reviews and began pushing for more pair-programming.

We also included more people than just the developers in the game. This time we chose to include the Quality Assurance team as well as the Product Management team. This was a little difficult because the pay chart that the other teams worked from were very different from that of the developers. Unfortunately, this meant that we had to adjust the amounts of money the other teams had to keep things even. Otherwise the other teams would lose out when the auction rolled around.

The enthusiasm with the developers was immediate this time. The instant we announced the start of the game, people were teasing and ribbing each other about preventing them from getting what they wanted. Even before we knew what the auction items were!

This iteration we also took pictures of each item and one of the guys created a small web-site on his computer where everyone could go see the prizes. This was more fun, since some of the guys kept spoofing the web site and putting photos of obviously fake items or doctored photographs of the other developers.

With all the visibility of the items, it was interesting to overhear people talking with spouses about prizes and see

them looking up actual values on the Internet.

12. The Second Auction

The second auction was much larger since we had more people. We brought in lunch again and had everyone gather in a much larger area than before.

We brought in the same fast-talking auctioneer, who had obviously been studying his stand up. He was much funnier than the first time, and those who were offended eventually got over it.

The auction itself was much louder and more rambunctious. People were yelling across the room and really taunting each other. It quickly became obvious that the auctioneer needed to take more control of the auction. The first time, we could take our time and allow people to contemplate whether to bid higher or not. This time, with many more people, we needed to keep things moving fast and our auctioneer picked up on that rather quickly. He started cutting short the bids and forcing people to make snap decisions instead of allowing them time to think.

We also observed similar behavior as the first auction with people conning each other out of their remaining cash. One developer even went so far as to offer an exchange rate for real cash just prior to the auction.

13. Observations from the Second Game

After playing the game twice, we were able to look at the differences and make some more mature observations.

One thing we noticed is that the game needs to focus on the incentive value and not on fixing a particular problem. Our focus initially was to fix the problem getting developers to adhere to a new process. While ultimately it did accomplish that, I think that focusing on the incentive value and allowing the problem to unfold itself was the right answer.

Another observation made by one of the managers is that you have to avoid predictability in the game. There needs to be some element of adventure and unknown to the game, otherwise there isn't enough excitement to keep interest over a long period. The game should also be different between iterations. It doesn't need to be drastically different, just enough to keep it exciting.

14. Reflection

We originally created the game to increase morale and lessen the resistance to change. What we actually achieved was more: spontaneous pairing, increased productivity, volunteer suggestions to more process changes, and overall more involvement in the process decisions.

Probably the biggest discovery that we made is that people basically love competition. If you can create competition that benefits the team and the project, and

provides rewards, then you have a good thing. The caution is not to have it too closely related to the project. If the competition involves the project itself, then it becomes possible for the project to suffer in places where someone suffers in the game. If the game is based on the process activities, the project continues untouched even though the process activities may not be entirely up to par. In an extreme case, one developer trying to “get one up” on another may cause waves in the game but not the project.

We have taken the competition aspect into other aspects of the team. For example, we have a small budget for training/conferences. There is not enough money for everyone to go to what they want so we have created a competition where team members can provide training to the other developers, study/research technologies outside of their normal duties, etc. Each activity provides a certain number of points. At the end of the year, the person with the most points gets to choose the conference they want to attend. We then proceed down the list allowing people to choose as long as it is within the budget. This is new so we will see how it works out at the end of the year!

Everyone likes to have fun and the every-day work environment should not be an exception. Any time you can make work fun, you should do it. Other departments in our division have taken notice and a few have played the game in their own way. Some have succeeded better than others, but none were failures. As a point of interest, we are currently in the middle of another iteration of the game and are going to have the auction in July.

I would recommend the game to anyone who is attempting to implement some large changes to their process and anticipating resistance to the change, or to a team that is less cohesive than they would like to be. In fact, I would recommend the game to anyone who just wants to keep a team alive and enjoying their work. We feel like the game brought the team closer together as well as closer to management. There seems to be less of an “us vs. them” attitude.

15.Thoughts For those who want to try it

- Provide many different ways to earn money.
- Create some “end game” competitions to allow someone who is behind to catch up.
- Have the game take some time. We ran it for a period of months for each iteration.
- Have some fun bonuses in the midst of the work. The small “games within the game” are fun and provide a break from the routine.
- Place the auction items in a visible place to promote the excitement.
- Share the task of purchasing the items among the managers/leaders. Discuss the prizes before hand so that ideas feed more ideas.
- Have inexpensive gag items.
- Plan the auction during lunch and provide the food. It

makes for a more enjoyable and less formal atmosphere.

- Get someone who is very dynamic to be the auctioneer. Someone who is comfortable in hamming it up in front of people is absolutely necessary.

16.Acknowledgments

Thanks to my family for their sacrifices while I explore new avenues. Thanks to the people I work with who have provided an environment rich with opportunity for learning and discovery. Thanks to the Salt Lake Agile Group for their ideas and encouragement, and especially Alistair Cockburn for his coaching.

17.References

[1] Scott Ambler, Process Patterns, Cambridge University Press / SIGS Books

[2] Phil Orbanes, Everything I know about business I learned from Monopoly, Harvard Business Review, Reprint R0203C