The Journey: Winning the Clone Wars Part 1

Posted At : October 31, 2012 12:21 PM | Posted By : Mark Kruger
Related Categories: Business Of Development

In my last post on this topic back in September, **Phase II - The Clone Wars**, I discussed the first phase of our business development. We talked about how I tried to duplicate my own skills and energies by hiring likeminded folks, and how this led to a lack of diversity and innovation. In this post we will pick up on some of the solutions to those issues. Let me say at the outset that some of these issues (founders syndrome for example) are systemic and require constant vigilance and an ongoing effort to resolve. After all, we didn't come up with this list overnight at Denny's and pop in the next morning with neat and tidy solutions to all of them. Some of the items on our list (the need for sales, the value of diversity, the importance of management, team building etc.) required some convincing and cajoling and even some hard knocks to move us in the right direction. But I can say that in spite of "peaks and valleys" (which was incidentally my nickname in high school) we are moving in the right direction. So let's talk about solutions for moving off of the clone model and to something more workable for a larger, team-oriented staff.

Blame Starts at the Top

Let's start with me - after all, no one is more susceptible to founders syndrome than a founder right? When evaluating company growth and moving beyond a small close knit staff, the first question I had to ask myself is *"Do I want this?"* Am I cut out for a larger business than 4 or 5 developers? And most importantly, am I ready to give up working *in* the business and can I be satisfied simply working *on* the business. Unless you are an MBA starting a business from scratch in a field in which you are disinterested, this is something every founder faces. In order to move past the clone phase I was going to have to engage as management and owner and *not* as a developer. I was going to have to stop coding full time on customer sites. I was going to have to relinquish control over various aspects of the company as it grew too large for me to wield the reigns by myself. I would have to "trust others with my baby" so to speak.

This is, quite frankly, still a weekly struggle. My team has to chastise me about jumping into areas where I don't belong. They would prefer if the only "direct" work I did was troubleshooting, coaching and mentoring and sales closings - leaving the day to day stuff all to them. Except in times of crisis I honor this approach. But it was hard leaving the emersion in code behind. Like most founders I started as a knowledge expert *doing* the stuff around which my company is organized - just like a master plumber hires other plumbers but continues to...uh...to plumb. Now my role is much different. Here's a typical day's log of work for me (I keep a small daily journal). This represents the individual tasks I accomplished on a typical day.

- Daily Check-in with exec staff.
- Transferred funds.
- Signed contract checks.
- Worked on a blog post.
- Found I had been walking around with my fly unzipped.
- Scheduled 2 customer meetings for tomorrow.
- Helped dev manager with FreePBX settings for remote extensions.
- Lunch with Developers.
- Wrote expense checks for travel.

- Hopped up and down for 3 minutes.
- Checked status of ACH payments.
- Notified customers regarding pending blocks of hours.
- Facilitated finding and delivering .NET code from a project 2 years ago per customer request.
- Discussed new migration project with PMs and located and forwarded an original plan from some months ago.
- Discovered I had lettuce stuck in my teeth.
- Worked with team members on a CF10 migration providing advice and consultation.
- Created Cert request for customer site (the person who usually does this was out).
- Participated in 2 topical meetings with exec staff.

The rest of the time (an hour or so out of a 9 hour day) was devoted to reading some blogs and doing some noodling regarding business planning.

As you see my day included just 4 tasks out of 18 that were in any way "technical" in nature - and I'm counting things like helping the dev manager with FreePBX settings. There are days when I have more technical work to do. I manage to work on our internal systems a few hours a week. But I am mostly doing tasks that facilitate current projects, handle employee issues and *look forward* to the next phase of my business. I can easily see a time in the near future when *all* of my tasks are those "looking forward" type of tasks.

Mourning and Nostalgia

As a founder with a love for technology I have had to come to grips with the stages of grief. This doesn't mean I wander around the office crying and asking why, but it does mean that I had to make a conscious effort to give up something I love for something else I loved and wanted. In fact, my choice was *do I love development more than developers*. Remember item 2 on my **3 attitudes** post? It is "Be a people person first". So I gave up actual development in favor of building a community of developers - something I approached with equal joy and satisfaction. Still, the days of coding full time are a loss I feel surprisingly deep. It's hard for the uninitiated to understand. I rarely dream in code any more. I don't have many of those "Aha!" moments where I stumble onto a solution by bearing down on it for hours at a time. I still get to load test, troubleshoot and tune servers - and that keeps me engaged in the CF world. But it is a loss I feel - an intentional sacrifice I had to make to allow my business to break out of the cloning phase.

The truth is that I am a terrific technologist and a great coder with great cognitive troubleshooting abilities. But I have been a mediocre business man. My chief accomplishment has been in not making egregious mistakes that sink the company, not being in a hurry to get rich and continuing to noodle my way through the minefield of an emerging business (as these post hopefully show). So taking on this different role *definitely* took me out of my comfort zone and I felt (still feel at times) out of sorts trying to figure out my next step. The biggest issue a founder has is trust. Trust doesn't come from *agreement* or even "like-mindedness" per se. It comes from an intentional

authority that is vested in folks you believe:

- Have the appropriate skill set.
- Will try to *do the right thing*.
- Will give you their best effort

Many founders are quick on the trigger with firing and hiring due to a lack of trust. Such founder's issues don't stem from evaluation of competence (presumably their employees are generally qualified) but rather from a sense that, when mistakes are made, the folks beneath must not being "doing the right thing" or "doing their best". There is also a sort of "I-would-not-have-made-that-mistake" hubris that goes along with being even a mediocre founder. Fortunately years of therapy and electric shock have helped the Muse be kinder and gentler. I recognize that I am a ginormous idiot some of the time and a somewhat smaller idiot the rest of the time. God has blessed me with a *terrific team* of developers and managers who deserve to be trusted. Being willing to fight that personal battle within myself and freely give of my trust has helped me in the transitions I face - though it is still a daily struggle.

Don't Let Your Developers Grow Up to be Cowboys

The second battle in the clone wars was to eliminate "cowboy development". By cowboy development I mean the following:

- Developers are free to make *all* decisions regarding a task with no appeal to a standard or guidelines.
- Developers are directly involved in deployments.
- Developers are responsible for unit testing, QA, and decisions on when a task is completed and ready to deploy.
- Developers have no one looking at their code with an eye toward improving it.
- Developers communicate directly with the stakeholder with no intermediary or third party involvement.

Now there are some *very* good cowboy developers (though it's hard to find a hat like Hoot Gibson's in the above pic). I would count the Muse among them I guess. If you give me a project I can usually find my way around all the nuances of that project, solve any roadblocks, and complete, test and deploy the project. Usually the result is relatively bug free and I work fast. Any "good" contractor (by which I mean a single individual sustaining several clients and satisfying all of them) is in some sense a cowboy developer right?

But such development relies on the individual judgment of a developer instead of relying on a *rules* based process where lists of details are vetted and confirmed. In a team environment everyone has to trust that the items they are working on fit into the larger picture - sometimes without having a full view of that larger picture. In addition, while I'm a good cowboy developer, my cowboyness highlights some deficiencies in my development process - like lack of collaboration, too much dependence on "smarts", too free-wheeling and seat-of-the-pants, not placing enough value on specificity, not placing enough value on testing and QA etc. Developers who were equally competent but without these deficiencies would actually have trouble

succeeding in our little home on the range as long as we were playing cowboy. Finally, cowboy developers have a ceiling with regard to the *size* of projects they can reasonably be expected to complete. If we wanted to grow and work on much larger applications we were going to have to solve this problem.

So one of the first changes we had to make is to add specific management controls to our system. This meant coding standards, testing, emphasis on collaboration, and project management for each team in our group of teams. It also meant a shift in development from the old boy method of setting up sites in a dev environment and coding directly against the server (and using source control as a sort of modified backup) to a formal SDLC of coding locally, committing to SVN and then automating a staging environment for QA and stakeholder input (we use Jenkins and ANT for this). This part of our "procedural protocols" has benefited us greatly. Developers on regular clients rarely have to log into dev or production environments any more. They can focus on their local dev environment. In fact, we created some custom Eclipse SVN templates that developers can use to view tracks and update our tracking system with hours and comments (and the file list off of the commit is added automatically as part of the notes). So a typical developer does not even need to leave Eclipse to do everything she or he needs to do.

We still have occasional cowboy moments to be sure - some of them even involve the Muse overstepping his bounds, but by and large the dev staff is aligned around a common set of development protocols that help us maintain our focus and eliminate unforced errors. Once our development was aligned, these core concepts found their way back into our hiring process where we start with a "cultural interview" - an interview that specifically teases out how well we believe a new developer will "fit" within our system. This has (we believe) made our hiring more effective and resulted in a higher quality staff going forward.

Diversity

When speaking of diversity I probably don't mean what you think I mean. We value people of all origins and walks of life and we never make gender or ethnicity an issue when hiring. But what I'm talking about here is *technical diversity*. This is something that is hard to achieve and (unfortunately) too many developers and IT people fail to see the value of it.

Technical staffs tend to mimic many of the features of tribal or band level societies. In such societies a set of rules or norms emerges over time and become institutionalized (usually with religious ritual). The origin of a given rule usually (though not always) has some basis is behavioral fact.

An alternate view or path is not just "a difference of opinion" but an affront to the gods and the tribe - causing excommunication. Once these dogmas are established they become very hard to change without revolution or sectarian splintering - even when the underlying premise is no longer valid. The upside of this is a unifying dynamic. The tribe sticks together, lives in community and helps each other survive. The downside is a propensity to war against neighboring tribes. Anyone outside the tribe is considered beneath contempt. There are only 2 groups, *us* (the tribe) and *them* (everyone not in the tribe). As an aside, this dynamic plays out in virtually all groups of common cause - democrats against republicans, evangelicals against the secular world, vegetarians against carnivores, the French against everyone else etc.

Like tribal religions, IT folks tend to coalesce around a technology and a certain point of view. Opinions on technical issues become institutionalized and then become subject to what political scientists call the "innate conservatism of political institutions" - meaning they build up strong defenses against change and become inflexible. They are no longer adaptable (think "US Congress"). Now settling on core technologies is a good thing and can lead to efficiencies, but it should never result in *us against them*. I've seen staff's where the use of the command line (to give an example) was a sort of "elite hallmark" of technical expertise and where folks who prefer a GUI interface were looked down upon. Other staffs eschew Windows for Apple or use Linux for everything. Apache users disdain IIS. Windows Admins see Linux users as little more than anarchists. But truthfully, command line users are not really very impressive and they miss out on some wonderful tools. Meanwhile GUI users sometimes bog down in clicks and screens. Windows 7 is a great product with a huge array of features and applications. The Apple "eco-system" is a unique integrated way to live a virtual life. Nothing in the Linux world comes close to the integration and network control of Active Directory. Nothing in the Windows world is as lean and efficient as a LAMP stack.

Even though the Muse can say all these things with some degree of authority, the various tribes that read my blog are putting on war paint and preparing salvos for the comment section as we speak. Just keep in mind when you react that way you are making my point for me :). Of course these tribal viewpoints have their origin in opinions that were at one time valid, and all of the underlying reasons for those opinions have partially or (in some cases) completely decayed. When assembling a high quality staff we are looking for the *non-tribal technologist*. This is not someone without preferences, but rather someone who can empathize with other points of view, has a high natural *curiosity* and an aptitude for grasping technical paradigms and learning them quickly.

For example, we need (and hire) ColdFusion developers who are familiar with frameworks. We work with FW/1, Mach-II, Model Glue, Fusebox etc. But we would probably *not* hire (we have learned our lesson) a developer who was forcefully *committed* to a particular framework. We are looking for innovative thinking, not dogmatic lobbyists or advocates. To give another example, we have learned to not hire someone who spends time in the interview complaining or denigrating a particular technology (Apple or Microsoft are the two most often mentioned). One very competent developer we hired was placed on a high quality team where he was required to run a dev environment with Windows 7, CF Builder and IIS. This developer could never overcome his tribal aversion to Windows. Even though the other 5 members of his team were very productive, this developer spent copious amounts of time wrangling with and complaining about the environment. Finally we had to replace him. In my view this (extremely bright) developer *could* have been successful on that team if he had adopted a different attitude. In other words, it was his *tribal attitude* and not his *aptitude* that was the genesis of his failure.

So diversity for CF Webtools means:

- An eclectic skill set.
- Enthusiasm for technologies without religious fervor.
- Curiosity and a "learning" lifestyle.
- Master of 1 to 3 technologies and a dabbler in many.
- A "knowledge sharing" mentality that desires to disperse both technical and

domain knowledge to team members.

In addition, we want our "aggregate brain" to cut a wide swath with regard to both technical and domain expertise. When we interview we often spend time looking at the ancillary skills a developer brings to the table. We want to know how a given developer will broaden our reach. This moves us away from the clone model. The more diverse our pool the more cross pollination takes place. Unlike cloning, a diverse staff with few tribal boundaries spurs innovation and a culture of effective learning and growth. We are not there yet, but we are moving decisively in the right direction.

Conclusion

Ok, so if you are reading this post you might be tempted to believe that we think we have arrived. That's far from true. Have we learned some valuable lessons? Yes of course. I hope that is clear. But we are definitely a work in progress and there are many things yet to learn and unlearn. In my next post (Winning the Clone Wars Part 2) I'll talk about changes in management and sales.

Finally, this post contains some references to technologies that some of you find near to your heart. My admonishment to you would be to avoid that visceral tribal reaction you have and the temptation to dive in and comment on your pet tech paradigm. This post is *not about technology*. It's about how we are trying to build something great around ideas and principles that include developers at the core. Please be respectful - you know the Muse loves ya! :)