# How To Keep Your Best Programmers

Posted on August 12, 2012 by Erik Dietrich

## Getting Philosophical

Given that I've just changed jobs, it isn't entirely surprising that I've had a lot of conversations recently about *why* I decided to do so. Generally when someone leaves a job, coworkers, managers, HR personnel, friends, and family are all interested in knowing why. Personally, I tend to give unsatisfying answers to this question, such as, "I wanted a better opportunity for career advancement," or, "I just thought it was time for a change." This is the corporate equivalent of "it's not you– it's me." When I give this sort of answer, I'm not being diplomatic or evasive. I give the answer because I don't really know, exactly.

Don't get me wrong. There are always organizational gripes or annoyances anywhere you go (or depart from), and it's always possible that someone will come along and say, "How would you like to make twice as much money doing the coolest work imaginable while working from home in your pajamas?" or that your current employer will say, "We're going to halve your pay, force you to do horrible grunt work, and send you to Antarctica to do it." It is certainly possible that I could have a specific reason for leaving, but that seems more the exception than the rule.

As a general practice, I like to examine my own motivations for things that I do. I think this is a good check to make sure that I'm being rational rather than impulsive or childish. So I applied this practice to my decision to move on and the result is the following post. Please note that this is a foreword explaining what got me thinking along these lines, and I generalized my opinion on my situation to the larger pool of software developers. That is, I'm not intending to say, "I'm the best and here's how someone can keep me." I consider my own programming talent level irrelevant to the post and prefer to think of myself as a competent and productive developer, distinguished by enthusiasm for learning and pride in my work. I don't view myself as a "rock star," and I generally view such prima donna self-evaluation to be counterproductive and silly.

## What Others Think

Some of my favorite blog posts that I've read in the last several years focus on the subject of developer turnover, and I think that these provide an excellent backdrop for this subject. The oldest one that I'll list, by Bruce Webster, is called "The Wetware Crisis: the Dead Sea Effect," and it coins an excellent term for a phenomenon with which we're all probably vaguely aware on either a conscious or subconscious level. The "Dead Sea Effect" is a description of some organizations' tendency to be so focused on retention that they inadvertently retain mediocre talent while driving better talent away:

*…what happens is that the more talented and effective IT engineers are the ones most likely to leave — to evaporate, if you will. They are the ones least likely to put up with the frequent stupidities and workplace problems that plague large organizations; they are also the ones most likely to have other opportunities that they can readily move to.*

*What tends to remain behind is the 'residue' — the least talented and effective IT engineers. They tend to be grateful they have a job and make fewer demands on management; even if they find the workplace unpleasant, they are the least likely to be able to find a job elsewhere. They tend to entrench themselves, becoming maintenance experts on critical systems, assuming responsibilities that no one else wants so that the organization can't afford to let them go.*

Bruce describes a paradigm in which the reason for talented people leaving will frequently be that they are tired of less talented people in positions of relative (and by default) authority telling them to do things–things that are "frequent stupidities." There is an actual inversion of the pecking order found in meritocracies, and this leads to a dysfunctional situation that the talented either avoid or else look to escape as quickly as possible.

Bruce’s post was largely an organizational perspective; he talked about why a lot of organizations wind up with an entrenched group of mediocre senior developers, principals, and managers without touching much on the motivation for the talented to leave beyond the “frequent stupidities” comment. Alex Papadimoulis from the Daily WTF elaborates on the motivation of the talented to leave:
*In virtually every job, there is a peak in the overall value (the ratio of productivity to cost) that an employee brings to his company. I call this the Value Apex.*

*On the first minute of the first day, an employee’s value is effectively zero. As that employee becomes acquainted with his new environment and begins to apply his skills and past experiences, his value quickly grows. This growth continues exponentially while the employee masters the business domain and shares his ideas with coworkers and management.*

*However, once an employee shares all of his external knowledge, learns all that there is to know about the business, and applies all of his past experiences, the growth stops. That employee, in that particular job, has become all that he can be. He has reached the value apex.*

*If that employee continues to work in the same job, his value will start to decline. What was once “fresh new ideas that we can’t implement today” become “the same old boring suggestions that we’re never going to do”. Prior solutions to similar problems are greeted with “yeah, we worked on that project, too” or simply dismissed as “that was five years ago, and we’ve all heard the story.” This leads towards a loss of self actualization which ends up chipping away at motivation.*

*Skilled developers understand this. Crossing the value apex often triggers an innate “probably time for me to move on” feeling and, after a while, leads towards inevitable resentment and an overall dislike of the job. Nothing – not even a team of on-site masseuses – can assuage this loss.*

*On the other hand, the unskilled tend to have a slightly different curve: Value Convergence. They eventually settle into a position of mediocrity and stay there indefinitely. The only reason their value does not decrease is because the vast amount of institutional knowledge they hoard and create.*

This is a little more nuanced and interesting than the simple meritocracy inversion causing the departure of skilled developers. Alex's explanation suggests that top programmers are only happy in jobs that provide value to them and jobs to which they provide *increasing* value. The best and brightest not only want to grow but also to feel that they are increasingly useful and valuable–indicative, I believe, of pride in one's work.

In an article written a few years later titled "Bored People Quit," Michael Lopp argues that boredom is the precursor to developers leaving:

*As I've reflected on the regrettable departures of folks I've managed, hindsight allows me to point to the moment the person changed. Whether it was a detected subtle change or an outright declaration of their boredom, there was a clear sign that the work sitting in front of them was no longer interesting. And I ignored my observation. I assumed it was insignificant. He's having a bad day. I assumed things would just get better. In reality, the boredom was a seed. What was "I'm bored" grew roots and became "I'm bored and why isn't anyone doing anything about it?" and sprouted "I'm bored, I told my boss, and he… did nothing," and finally bloomed into "I don't want to work at a place where they don't care if I'm bored."*

*I think of boredom as a clock. Every second that someone on my team is bored, a second passes on this clock. After some aggregated amount of seconds that varies for every person, they look at the time, throw up their arms, and quit.*

This theme of motivation focuses more on Alex's "value provided to the employee" than "value that employee provides," but it could certainly be argued that it includes both. Boredom implies that the developer gets little out of the task and that the perceived value that he or she is providing is low. But, beyond "value apex" considerations, bored developers have the more mundane problem of not being engaged or enjoying their work on a day to day basis.

## What's the Common Thread?

I'm going to discount obvious reasons for leaving, such as hostile work environment, below-market pay, reduction of benefits/salary, etc., as no-brainers and focus on things that drive talented developers away. So far, we've seen some very compelling words from a handful of people that roughly outline three motivations for departure:

- Frustration with the inversion of meritocracy (“organization stupidities”)
- Diminishing returns in mutual value of the work between programmer and organization
- Simple boredom

To this list I’m going to add a few more things that were either implied in the articles above or that I’ve experienced myself or heard from coworkers:

- Perception that current project is futile/destined for failure accompanied by organizational powerlessness to stop it
- Lack of a mentor or anyone from whom much learning was possible
- Promotions a matter of time rather than merit
- No obvious path to advancement
- Fear of being pigeon-holed into unmarketable technology
- Red-tape organizational bureaucracy mutes positive impact that anyone can have
- Lack of creative freedom and creative control (aka “micromanaging”)
- Basic philosophical differences with majority of coworkers

Looking at this list, a number of these are specific instances of the points made by Bruce, Alex and Michael, so they aren’t necessarily advancements of the topic per se, though you might nod along with them and want to add some of your own to the list (and if you have some you want to add, feel free to comment). But where things get a little more interesting is that pretty much all of them, *including* the ones from the linked articles, fall into a desire for autonomy, mastery, or purpose. For some background, check out this video from RSA Animate. The video is *great* watching, but if you haven’t the time, the gist of it is that humans are not motivated economically toward self-actualization (as widely believed) but are instead driven by these three motivating factors: the desire to control one’s own work, the desire to get better at things, and the desire to work toward some goal beyond showing up for 40 hours per week and collecting a paycheck.

Frustration with organizational stupidity is usually the result of a lack of autonomy and the perception of no discernible purpose. Alex’s value apex is reached when mastery and purpose wane as motivations, and boredom with a job can quite certainly result from a lack of any of the three RSA needs being met. But rather than sum up the symptoms with these three motivating factors, I’m going to roll it all into one. You can keep your good developers by making sure they have a compelling narrative as employees.

## Guaranteeing the Narrative

Bad or mediocre developers are those who are generally resigned or checked out. They often have no desire for mastery, no sense of purpose, and no interest in autonomy because they've given up on those things as real possibilities and have essentially struck a bad economic bargain with the organization, pay amount notwithstanding. That is, they give up on self-actualization in exchange for a company paying a mortgage, a few car payments, and a set of utilities for them. I've heard a friend of mine call this "golden handcuffs." They have a pre-defined narrative at work: "I work for this company because repo-men will eventually show up if I don't." These aren't necessarily bad or unproductive employees, but they're pretty unlikely to be your best and brightest, and you can be assured that they will tend to put forth the minimum amount of effort necessary to hold up their end of the bad bargain.

These workers are easy to keep because that is their default state of affairs. Going out and finding another job is not the minimum effort required to pay the bills, so they won't do it. They are Bruce's "residue" and they will tend to stick around and earn obligatory promotions and pay increases by default, and, unchecked, they will eventually sabotage the RSA needs of other, newer developers on the team and thus either convert them or drive them off. The narrative that you offer them is, "Stick around, and every five years we'll give you a promotion and a silver-plated watch." They take it, considering the promotion and the watch to be gravy.

But when you offer that same narrative to ambitious, passionate, and talented developers, they leave. They grow bored, and bored people quit. They refuse to tolerate that organizational stupidity, and they evaporate. They look for "up or out," and, realizing that "out" is much quicker and more appealing, they change their narrative on their own to "So long, suckers!"

You need to offer your talented developers a more appealing narrative if you want them to stay. Make sure that you take them aside and reaffirm that narrative to them frequently. And make sure the narrative is deterministic in that their own actions allow them to move toward one of the goals. Here are some narratives that might keep developers around:

- "If you implement feature X on or ahead of schedule, we will promote you."
- "With the work that we're giving you over the next few months, you're going to become the foremost NoSQL expert in our organization."

- “We recognize that you have a lot of respect for Bob’s Ruby work, so we’re putting you on a project with him to serve as your mentor so that you can learn from him and get to his level.”
- “We’re building an accounting package that’s critical to our business, and you are going to be solely responsible for the security and logging portions of it.”
- “If your work on project Y keeps going well, we’re going to allow you to choose your next assignment based on which language you’re most interested in using/learning.”

Notice that these narratives all appeal to autonomy/mastery/purpose in various ways. Rather than dangling financial or power incentives in front of the developers, the incentives are all things like career advancement/recognition, increased autonomy, opportunities to learn and practice new things, the feeling of satisfaction you get from knowing that your work matters, etc.

And once you’ve given them some narratives, ask them what they want their own to be. In other words, “we’ll give you more responsibility for doing a good job” is a good narrative, but it may not be the one that the developer in question envisions. It may not always be possible to give the person exactly what he or she wants, but at least knowing what it is may lead to attractive compromises or alternate ideas. A new team member who says, “I want to be the department’s principal architect” may have his head in the clouds a bit, but you might be able to find a small, one-man project and say, “start by architecting this and we’ll take it from there.”

At any point, both you and the developers on your team should know their narratives. This ensures that they aren’t just periodic, feel-good measures–Michael’s “diving saves”–but constant points of job satisfaction and purpose. The developers’ employment is a constant journey that’s going somewhere, rather than a Sisyphean situation where they’re running out the clock until retirement. With this approach, you might even find that you can coax a narrative out of some “residue” employees and reignite some interest and productivity. Or perhaps defining a narrative will lead you both to realize that they are “residue” because they’ve been miscast in the first place and there are more suitable things than programming they could be doing.

## Conclusion

The narratives that you define may not be perfect, but they’ll at least be a start. Don’t omit them, don’t let them atrophy and, whatever you do, don’t let an inverted meritocracy–the “residue”–interfere with the narrative of a rising star or top performer. That will catapult your group into a vicious feedback loop. Work on the narratives with the developers and refine them over the course of time. Get feedback on how the narratives are progressing and update them as needed.

Alex thinks that departure from organizations is inevitable, and that may be true, but I don’t know that I fully agree. I think that as long as talented employees have a narrative and some aspirations, their value apex need not level off. This is especially true at, say, consulting firms where new domains and ad-hoc organization models are the norm rather than the exception. But what I would take from Alex’s post is the perhaps radical idea that it is okay if the talented developer narrative doesn’t necessarily involve the company in five or ten years. That’s fine. It allows for replacement planning and general, mutual growth. Whatever the narrative may be, mark progress toward it, refine it, and make sure that your developers are working with and toward autonomy, mastery, and purpose.