developer.*
The Independent Magazine for Software Professionals

Software Team Turnover:
Why Developers Leave (And What You Can Do About It)

by Aaron Reed

Unplanned turnover hurts—especially when you lose top people. Good developers are a rare breed. If you've ever hired a new software developer, you know how difficult it is to find people that are competent.

Good developers that work well with your existing team of developers are even harder to find. Even when you find a person with the right skills, personality conflicts and other issues can arise to complicate the team dynamics and jeopardize the team's cohesion. When a good team has finally been assembled, it's imperative to keep them together and working well.

In addition to the loss of team cohesion, the organization loses the intangible knowledge that good developers take with them when they leave. Even with good documentation, developers possess undocumented knowledge about the product, the domain, and the designs that is essential to the project's completion. The fact that most development efforts also suffer from less than adequate documentation compounds the impact that the loss a developer's product knowledge has on the overall outcome.

In addition to product knowledge, key developers have indispensable knowledge of the organization's processes, development tools, source code control, coding standards, and more. All of this adds up to a very costly void that any company needs to fill when key developers jump ship.

Far too frequently a manager or executive will get caught up in the mindset that the salespeople are the most important cogs in the machine. After all, the sales people are the ones bringing in the money, right? Well, yes and no. Sales people are important, and a good sales staff is essential, but if you have no developers to build the product, what will the sales people sell?

It sounds simple, but I've been involved with far too many companies where sales people are treated like gold and developers are treated more like rusty iron. Whether your organization is sales-oriented or not, if you depend on your software developers for success, you might be missing opportunities because of unplanned turnover that could have been avoided.

So the question then becomes: How can you avoid developer turnover and keep key team members on staff for lengthy periods of time? Well, there's no simple answer, but I've got some ideas to share with you. All situations are different, and there are countless reasons why people may or may not seek a change. I'll discuss the top three reasons why developers tend to look for greener pastures. Considering these three areas and making some adjustments may help you hold on to key pieces of your puzzle.

Money

Easily one of the biggest issues that causes developers to look elsewhere is probably the first issue which causes managers to cringe. That's right, most developers are just like everybody else in that they too exclaim the immortal words of Jerry Maguire, "Show me the money!"

Unfortunately, most developers are painfully aware that the best way to get a raise is to simply get a new job. A developer might be happy in all aspects of her job, but if she knows that her company is paying her below market rates, that fact might outweigh all the good. There are few companies that are willing to give a good, yet underpaid, developer a raise that will exceed what the developer could get by jumping ship and landing in a new job elsewhere.

I know what you're thinking. You can't possibly afford to pay all your developers enough money to compete with any offer that may be out there on the horizon. So what is the solution then? It's a good idea to ensure that all development team members are paid a competitive rate, but the key is to identify a small portion of your development staff as people who you decide to invest in long term, and to take care of them in a way that will make them never want to leave for monetary reasons.

There are numerous ways to let a developer know that you are interested in working with him long term, and not all of them involve monetary rewards. Any kind of additional "perk" goes a long way towards making people happy. These can involve non-liquid monetary assets such as stock options as well as perks like telecommuting, free lunches/snacks/sodas, good hardware, other gadgets (cell phones, PDAs) and so forth.

Keep in mind that many developers are “geeks” and often a cool PDA or a sweet workstation can go a long ways in this department.

There is a catch here, however. While these kinds of incentives can go a long way towards pleasing employees, nothing can harm morale faster than broken promises. That is, promising a bonus is great, but if the money isn’t available, or the rules are changed midstream, and the developers end up not getting the promised bonus, it does far more harm than never having promised a bonus in the first place. Make sure that whatever you commit to is feasible and that you follow through with the promise.

Morale

If a good developer leaves the company for a reason that doesn’t directly involve money, it is often some type of morale problem—one that may be more widespread than you think. If people aren’t happy where they work, they will often leave for equal—or even less—pay elsewhere.

There are a few key things, especially when dealing with software developers, that can go a long ways towards keeping people happy and in turn, keeping them on your payroll.

I believe the biggest morale-related factor that causes good developers to look elsewhere boils down to the need for a new challenge. Part of the problem with keeping a developer challenged is inherent in the way software projects are run. For example, developers tend to learn on two different curves when landing a new development job which are pictured in Figure 1 below.

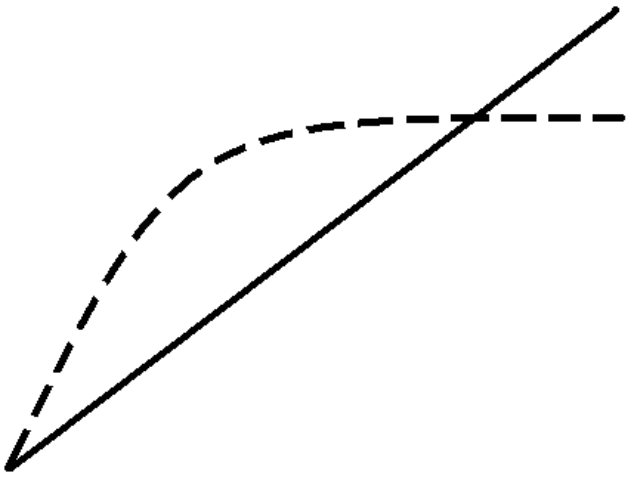

Figure 1: Developer learning curves.

The dotted line represents the progression a developer's rate of learning and enhancing his or her technical skills. When first hired, a developer learns at a rapid rate. This is due to working with new people, new processes, new tools (possibly new development languages), working with existing architectures, and so on. At a certain point, that learning curve begins to level out because the developer becomes familiar with the work, the tools, the languages, etc. At this stage the work can become a bit more routine and monotonous.

The solid line represents the rate at which the developer acquires domain knowledge while working on a given project. For example, if a developer starts a new job in the public library domain, that developer will begin learning more and more about the operation and nuances of public libraries. This curve tends to not level out as much or possibly not at all due to the fact that even though a software project may be released and completed, future research is spent on finding out ways to improve library operations and other user-related features that could be a selling point for the product.

As you can see from the graph, there is a certain point at which a developer starts learning more about the domain and not learning much, if anything, new technically. For many developers this may not be an issue, but for other developers this can result in a loss of interest in the company, the product, or their job as a whole.

A developer may be seeking to improve their skills, and not care as much about domain specific knowledge. If that is the case, this is the point at which a developer can become dissatisfied with his job and look on to bigger and better horizons.

The development of this situation is natural, but detecting it and doing something about it can be difficult. Get to know each of your talented developers individually and find out their career goals and desires. Each person is different, but a general theme you will encounter is that you need to keep your good developers technically challenged. Actually keeping them challenged is the difficult part. How to best go about it really depends on the individual developer—what challenges they seek and what motivates them individually.

By getting to know your developers, you will learn numerous ways that you can try to keep the technology learning curve from leveling out. All developers are different and different things inspire and motivate them. Find out what each person's motivating factor is, and then keep your good developers where they want to be. You may even find that some of your best developers will actually prefer working on an existing project with familiar code as opposed to working on a new project with new technologies. Try to keep your good developers working on projects they enjoy—whether that be an existing project a

person has worked on for a long time, or a brand new system that needs to be designed, or something in between.

Moving developers to new projects can be difficult because as developers build up domain and project knowledge, your first instinct is to keep them on that project and hire new developers to work on new projects. While that sounds like a good idea, it may or may not be the best solution for your developers and/or your company. What often happens is employers will keep proven developers on existing projects and hire unproven ones to spearhead new projects. While this may make sense from a business perspective in the short term, in the longer term—after the unproven new hire fails to develop a good product and the proven developer quits from boredom—it may not turn out so well.

When changing projects is not an option, there are also a number of other ways to allow developers to flex their brains now and then. These can include offering training opportunities to train people on the latest technical tools, sending people to technical conferences or workshops, offering tuition reimbursement to allow developers to continue their educations, giving good developers the go-ahead to analyze and re-work existing architectures, and so on. Be creative and let developers know that they are allowed and encouraged to further their own careers and not just your projects.

The bottom line here is to identify your good developers, the ones that you want to have around for a long time, and find out where they want to be and what they want to be doing. Work with them to make them happy and to keep the company moving forward. Even when you're not able to take the actions you'd like to take, simply pulling a developer aside and taking the time to find out what makes him or her happy goes a long way, in and of itself, towards building good morale.

Burnout

Finally, I'll address the third major reason why developers tend to switch jobs: burnout. Software development is a tedious business and a difficult task for all involved. As projects near completion, work life is full of meetings, disorganization, overtime, and stress. Even the standard day for a good developer is full of heavy thinking, problem solving, frustrating compilation errors, and the like. Software development is not your standard "sit at your desk" job. It's the equivalent of taking a difficult test 8 hours a day, 5 days a week (plus overtime).

Burnout is a real issue and something that faces nearly every developer at some time in their career. Often the thought is to change careers completely and get away from it all. Some look to management positions, hoping for less stress. More often than not, however, the "solution" is to look elsewhere for a new job thinking the immortal words, "it can't possibly be this bad over there."

The biggest thing that managers need to be able to do in order to prevent burnout is to allow their employees to have fun. This is a very hard thing for a lot of development managers to do. Taking 15 minutes to a half hour out of the day to have fun will seem superfluous to many. But there is so much that can be accomplished in a simple activity that may last only as long as your average smoke break.

A team that has fun together is much more likely to work together well, enjoy working with each other and help each other. Job satisfaction is increased when there is something other than work to look forward to as well. The mental drain of a tough day of debugging can severely be lightened by a few minutes of entertainment.

What kinds of activities am I talking about? That's up to you. People are different and different things work for different people so there isn't any one clear answer to that question. Many developers are computer gaming geeks as well so a quick match of Warcraft III or Unreal Tournament can serve to refresh a developer almost as well as a good nights sleep. Others don't enjoy PC gaming, but will welcome other things, like a quick game of HORSE on the Nerf basketball hoop in the hallway.

One company set aside a few minutes every Friday morning for miniature golf. Every person was responsible for setting up a miniature golf hole in their cubicle and then the team played a round of golf. Another company had a racing course setup in an unused conference room for Friday office chair races. Some companies have parties and luncheons to raise employee morale and give employees and occasional break. Be creative and come up with something that will be entertaining for the team members while also being constructive toward building team morale and keeping everybody energized for the work ahead.

Keep in mind that it is far better to offer an activity that costs less and free to employees than to offer something more lavish and charge employees an admission fee. For a team morale building exercise, nothing slaps you in the face as much as a money collector at the front door. Imagine being that one employee that is just making it paycheck to paycheck and is then excluded from work activities because of the cover charge. Not many things can destroy morale as quickly as that.

Conclusion

There are many other reasons why developers would lean towards moving on to greener pastures, but I've touched on some of the biggest ones. What this all really boils down to is managing a public relations campaign within your own organization in an effort to win over (and keep) your employees.

However, before looking to big, broad, expensive gestures and top-down morale improvement initiatives, look at the simple things first. Things like getting to know your employees personally, using common sense in operations, and finding out what employees want from you and your company can go a long, long way towards helping you keep good people working hard in your organization.

###

Aaron Reed is an Assistant Professor at Northface University in South Jordan, Utah. He holds a BS in Computer Science from Weber State University and a Microsoft MCP certification. Aaron has worked in the software development industry for 12 years in positions ranging from entry-level developer to vice-president of development at several companies. He has worked in .NET technologies since the first beta was released. When he isn't reading up on the latest development technologies, Aaron loves spending time with his wife and three children.