volume 6 issue 12

the informatizer™

The Software Industry Cheat Sheet
Published by Stout Systems
StoutSystems.com
877.663.0877

twitter.com/stoutsystems
linkd.in/stoutsystems

4 Areas That Attract and Retain Great Software Development Talent (from a Developer's Perspective)

PAGE 1

Handling Delicate Interview Questions

Have you thought about how to answer questions about previous jobs?

PAGE 9

STOUT® Fueled by the most powerful technology available: Human Intelligence.®

4 Areas That Attract and Retain Great Software Development Talent

(from a Developer's Perspective)

by Dave Sweeton

Go to any technical user group meeting and you are likely to find many announcements of jobs—but few developers who are looking for work. Something has definitely shifted in the supply and demand equation, and it's developers who are in the driver's seat now. They are able to change jobs more easily than at any time since before the last recession, and they frequently field multiple offers while doing so.

So what can you as an employer or manager do to recruit and retain good developers? There are a number of things. While it may not be possible to do all of them, it certainly would be possible to select a handful of them and institute them as quickly as possible.

#1 Technology Issues

Developers are keenly aware of how fast technology is shifting. They know if they don't keep up, they become dinosaurs. You don't want them to feel like they have to go elsewhere or risk becoming unemployable. Furthermore, being stuck on old technology or a mess of a codebase is boring and frustrating.

Legacy projects. Legacy codebases are a fact of life, but be aware that dated technology gets more expensive to support. Working only on out-of-date platforms is bad for the current programmer and can be a real stumbling block for hiring new programmers. Trying to find a candidate to take on VB6, MFC or classic ASP is a real challenge. Even ASP.Net WebForms is considered too out-of-date to touch by some candidates. For projects that are under active development, try to migrate and improve your technology stack to avoid getting trapped.

New Technologies/Libraries. Allow your team to use new technologies or libraries on existing projects, but only where it's appropriate and justified! You want to avoid having a kitchen sink of technologies or having libraries that are only around to justify a programmer's whim. Maybe try prototyping with a library before using it in the production code base, or just plain giving your programmers time allotted for investigation and "play." The skills they gain while investigating will likely lead to improvements in your projects.

Refactoring. Allow your developers time to refactor and pay down technical debt. Working in a codebase that grows worse with no end in sight is demoralizing. If you give your developers the time they need to produce quality code, everyone benefits. Make refactoring a task called out clearly in your project plans and don't allow it to be so low priority that it never gets done.

Challenges. Provide fun projects with new and interesting challenges. Okay, every project has some boring and menial aspects, but there has to be a balance of tasks where there's interesting work to be done. Every developer's definition of "interesting" varies of course, so ask!

Training. Invest in your developers and support them in getting training or attending conferences.

Productivity. Make sure you provide a fast enough computer and whatever productivity software they want. Investing in both of those pays dividends for the company and the developer.

Automation. If your project has boring tasks, encourage your team to find automated solutions. The most common examples would be automating builds, releases or deployments. This requires a time investment, but it's another one that is win-win.

Choice. If your team has a preference for a particular flavor of source control or issue tracker, and that choice meets your requirements, let them use what they want.

#2 Culture Issues

Developers suffer from being misunderstood. Very few outside of the software development team understand what is involved in writing quality code. A feature may seem like it should take ten minutes to implement, when it's actually a solid week's worth of work. If developers are constantly being beat up, they grow weary. Some friction, some misunderstanding, some pressure—these are fine once in a while. But a culture where that's the norm will surely drive developers away. Here are some things to consider:

Schedules. Have realistic schedules that are driven by the amount of work that needs to be done, not a calendar date that was chosen arbitrarily. If you do have a calendar deadline, cut the feature list down to something that is achievable by that date.

Estimates. Use estimates correctly. Don't ask your developers for ballparks and then treat them like estimates. Don't use estimates as a whip to punish your developers. Instead, figure out why the estimates were wrong and work with the developers to improve them in the future.

Development Time. Make sure your developers have lots of development time. Developers seem to be particularly sensitive to time "wasted" by inefficient or unnecessary meetings.

Interruptions. Support your developers in having blocks of time they can work without interruption, like ignoring emails and phone calls, or posting a sign/status that says "busy, no interruptions please." No interruptions means no interruptions (unless the building is actually on fire). Perhaps establish fixed time blocks where you can't schedule meetings, like no meetings before noon, or no meetings on Fridays.

Overtime. Limit the amount of overtime developers are required to work. Some developers welcome overtime but if overtime is required as the norm, it's usually a management failure.

#3 Integration Issues

On top of being misunderstood by the non-technical stakeholders, developers are often treated like pariahs—never permitted to mingle with the very people they are writing code for. Your developers need to be integrated into the larger ecosystem, interacting with end users, clients, customers, and stakeholders.

They need to hear firsthand about how their work is going to be used. They need to hear firsthand about frustrations, irritations, hopes, and dreams concerning their products. Filtering information to them is not the right action to take. And if you're worried about it, developers can be coaxed into communicating less technically so that these interactions are fruitful.

Participation. Have developers participate in requirements gathering and design tasks. If they hear the requirements from the horse's mouth, they will have a better idea of the real needs.

Requirements. When direct interaction is not possible, give developers good requirements. Nothing is more frustrating than developing something and having it rejected because of unstated or wrong requirements.

Creativity. Provide room for developer initiative and creativity. Being treated like an interchangeable "code-monkey" and told exactly what to do won't work for many developers.

Acknowledgements. Make certain developers hear about any kudos from your users. Specific praise from end users can really increase job satisfaction.

#4 Stand Out From the Crowd

There are a handful of things that developers really appreciate. If you can offer them, you should.

Environment. Provide a good work space and environment. For some developers this means quiet and private. For others a noisy bullpen may be ideal. If you can support both personality types, you'll be ahead of the game. Bullpen plus quiet rooms is one approach. Quiet workspaces plus great collaboration rooms is another. Either way, they will need a proper desk and chair with good ergonomics.

Flexibility. Allow for flexible work time. And consider offering remote/telecommute work. Everyone from every industry appreciates flexible work schedules, but some developers don't hit their most productive mental state until well after 5pm. Working remotely (occasionally or in large part), can be a huge productivity and happiness boost for some developers. Others will go crazy with the isolation or waste their time on distractions!

Remuneration. Lastly, but not least, pay well! Developers care about all aspects of the job, but pay is still as big an issue for us as anyone. Make sure you are paying salaries that are on par with similar jobs in your area. Your developers, with their knowledge of your business and codebase, should be worth more to you than hiring (and training) a new person with equivalent skills.

Hopefully from this long list of suggestions you will find a few that you can immediately implement to the benefit of your team. If you want to retain your best talent, making improvements on their behalf is a great way to show your concern and care.

Dave Sweeton is Stout's Chief Technologist. He began his career as a software developer in the mid-90s. Dave is responsible for numerous installations at major corporations throughout the United States.

Tight software budget?

At Stout, we'll work *with* you to find the software solutions that fit you best—both technically and financially.

The unknowns, the
undefineds, and the risks.

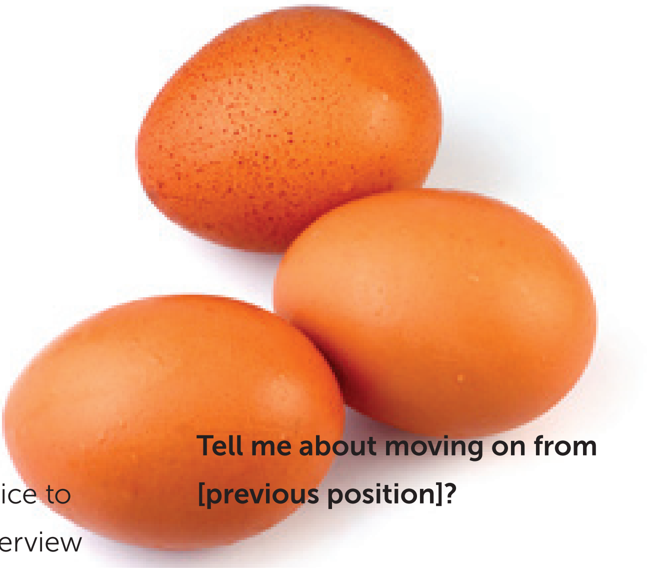
No, it’s not the name of
a band. It’s the day-to-day
of software development.

And Stout has done it
for over 20 years.
We can handle it.

Call Now
877.663.0877
StoutSystems.com

Handling Delicate Interview Questions

by Brian Skory



We've all heard the advice to be prepared for any interview question that comes your way—but let's be honest, we can't think through *every* question that could be asked. That said, we should at least give some thought to the questions that are *likely* to come our way—particularly those pesky "take you by surprise questions" that have the potential to catch you off guard. Answer it the right way and you can practically feel the interviewer's satisfaction with your response—but screw it up and you might as well shut your folio right then, thank the interviewers for their time, and walk yourself out the door. Here are some examples of those questions and the thought process behind them.

Tell me about moving on from [previous position]?

The interviewer is looking to see whether or not you left under favorable or unfavorable circumstances. Did you outgrow the position, and the company had nothing better to offer? Was your work shipped overseas? These would be acceptable responses. On the other hand, being caught off-guard about leaving because of something unfavorable (terminated for non-performance, couldn't get along with your boss, etc.) and stumbling to come up with an acceptable explanation will surely set off alarm bells. Be prepared to account for why you moved on from each position listed on your resume, and consider how your reasoning will be perceived from the hiring manager's perspective.

What could you have done to improve your performance at [previous position]?

The interviewer is simply checking to see if you are a process improvement-minded individual. If you are, it will be obvious that you have given this some thought and will be able to offer up something such as, "In retrospect, had I taken a bit of time to brush up on my Perl scripting, I would have been able to perform certain tasks much more efficiently." Stumble on this answer, and you risk being perceived as someone who doesn't particularly have improvement on their mind.

What would you consider one of your weaknesses to be?

This remains an ever-popular interview question, but again, if you are unprepared to answer it, the first thing that comes to mind could be the thing that kills the interview. No interviewer is expecting you to spill your guts or reveal your darkest secrets. Instead, they are looking to see if you've given this some thought—and if so, if you've been able to frame it with a positive outcome. An appropriate response would be, "I'm not much of a socializer at work, but I've been working on that so I don't come across as unfriendly." This shows that a candidate has thought about their weaknesses as well as potential solutions, and it puts the candidate in a much better light than a response such as, "I spend too much time at work checking on my Amazon storefront."

With a few minutes of forethought and mental rehearsal, you'll be able to put your best foot forward and maximize your chance of being the candidate of choice.

Brian Skory is Stout's Technical Talent Manager. He began his technical career 20 years ago and the key to his success has been his consistent ability to find and hire the best talent.

Software team missing a few key players?

We have the tech talent you need.

Contact Stout Today

StoutSystems.com

877.663.0877