How I Hire Programmers

November 29, 2009

Original link

Español

There are three questions you have when you're hiring a programmer (or anyone, for that matter): Are they smart? Can they get stuff done? Can you work with them? Someone who's smart but doesn't get stuff done should be your friend, not your employee. You can talk your problems over with them while they procrastinate on their actual job. Someone who gets stuff done but isn't smart is inefficient: non-smart people get stuff done by doing it the hard way and working with them is slow and frustrating. Someone you can't work with, you can't work with.

The traditional programmer hiring process consists of: a) reading a resume, b) asking some hard questions on the phone, and c) giving them a programming problem in person. I think this is a terrible system for hiring people. You learn very little from a resume and people get real nervous when you ask them tough questions in an interview. Programming isn't typically a job done under pressure, so seeing how people perform when nervous is pretty useless. And the interview questions usually asked seem chosen just to be cruel. I think I'm a pretty good programmer, but I've never passed one of these interviews and I doubt I ever could.

So when I hire people, I just try to answer the three questions. To find out if they can get stuff done, I just ask what they've done. If someone can actually get stuff done they should have done so by now. It's hard to be a good programmer without some previous experience and these days anyone can get some experience by starting or contributing to a free software project. So I just request a code sample and a demo and see whether it looks good. You learn an enormous amount really quickly, because you're not watching them answer a contrived interview question, you're seeing their actual production code. Is it concise? clear? elegant? usable? Is it something you'd want in your product?

To find out whether someone's smart, I just have a casual conversation with them. I do everything I can to take off any pressure off: I meet at a cafe, I make it clear it's not an interview, I do my best to be casual and friendly. Under no circumstances do I ask them any standard "interview questions" — I just chat with them like I would with someone I met at a party. (If you ask people at parties to name their greatest strengths and weaknesses or to estimate the number of piano tuners in Chicago, you've got bigger problems.) I think it's pretty easy to tell whether someone's smart in casual conversation. I constantly make judgments about whether people I meet are smart, just like I constantly make judgments about whether people I see are attractive.

But if I had to write down what it is that makes someone seem smart, I'd emphasize three things. First, do they know stuff? Ask them what they've been thinking about and probe them about it. Do they seem to understand it in detail? Can they explain it clearly? (Clear explanations are a sign of genuine understanding.) Do they know stuff about the subject that you don't?

Second, are they curious? Do they reciprocate by asking questions about you? Are they genuinely interested or just being polite? Do they ask follow-up questions about what you're saying? Do their questions make you think?

Third, do they learn? At some point in the conversation, you'll probably be explaining something to them. Do they actually understand it or do they just nod and smile? There are people who know stuff about some small area but aren't curious about others. And there are people who are curious but don't learn, they ask lots of questions but don't really listen. You want someone who does all three.

Finally, I figure out whether I can work with someone just by hanging out with them for a bit. Many brilliant people can seem delightful in a one-hour conversation, but their eccentricities become grating after a couple hours. So after you're done chatting, invite them along for a meal with the rest of the team or a game at the office. Again, keep things as casual as possible. The point is just to see whether they get on your nerves.

If all that looks good and I'm ready to hire someone, there's a final sanity check to make sure I haven't been fooled somehow: I ask them to do part of the job. Usually this means picking some fairly separable piece we need and asking them to write it. (If you really insist on seeing someone working under pressure, give them a deadline.) If necessary, you can offer to pay them for the work, but I find most programmers don't mind being given a small task like this as long as they can open source the work when they're done. This test doesn't work on its own, but if someone's passed the first three parts, it should be enough to prove they didn't trick you, they can actually do the work.

(I've known some people who say "OK, well why don't we try hiring you for a month and see how it goes." This doesn't seem to work. If you can't make up your mind after a small project you also can't make it up after a month and you end up hiring people who aren't good enough. Better to just say no and err on the side of getting better people.)

I'm fairly happy with this method. When I've skipped parts, I've ended up with bad hires who eventually had to be let go. But when I've followed it, I've ended up with people I like so much so that I actually feel bad I don't get to work with them anymore. I'm amazed that so many companies use such silly hiring methods instead.