by daniel chudnov
Information Technology Specialist
Office of Strategic Initiatives
Library of Congress

Practical Geek-Keeping, or, How to Hire—And Keep—Good Technical Staff

NOBODY LIKES TO INVEST TIME AND ENERGY ON A SUBSTANTIAL PROJECT ONLY TO SEE IT THROWN AWAY, IGNORED, OR FORGOTTEN FOR SOME HIGHER PRIORITY SYSTEM.

Recently, I was giving a talk about free software and open source in libraries at a state library consortium meeting. The attendees of the meeting included many library administrators. One of the questions I received after the talk stumped me. Although I came up with one good answer I'm still confident about and share below, it's taken me weeks to come up with more helpful answers to this question, for which many answers are possible. The question is, "How do you hire and manage technical staff whose work you don't understand?"

Communication in the Hiring Decision

A common answer I've heard from managers before, and the one I had for that person who asked, is that good communication skills are first and foremost. If you don't have good communication skills, no other qualifications or recommendations should matter. And I'm not talking about an awkward turn of a phrase in a cover letter or a clear indication that your applicant's native language is different from your own, however fluent they may be in your native language—it's the whole package that matters. The ability to express complicated ideas clearly and concisely, strong listening behaviors that aren't exaggerated or contrived, sustained attention to listeners while you speak, and a good memory for what others said earlier in a conversation are noticeable skills that can be evident during the briefest of interviews. And an absence of these skills can be evident in the first few moments after meeting somebody new.

The best interviews I've been involved in (whether as the applicant or an interviewer) are the interviews in which I've learned something about the nature of the work required of the applicant. It might be an explanation of a systems protocol, a lesson learned from how a past project evolved, or even just a small detail about an institutional process that might be easily overlooked but has an important impact. Whatever it is you've learned, though, when you have that moment of realization that, "I've learned something here today," you can look around the room and be confident that everyone there will probably be able to work well together because you're already learning from each other. And if you're already learning from each other, it's almost always the case that you're communicating well with each other. Remove good communications from that scenario and that learning might not happen so readily, if at all.

One good test of the ability of an applicant who is more technical than you is to ask for an explanation of a system he or she developed or an interesting technical problem he or she solved recently. If you indicate that the technical details are a bit beyond you, can the person modulate the explanation usefully, providing

missing background without getting lost in the details? Or does the applicant grow impatient and shrug it off? If you're going to work with this person, he or she needs to understand that there is information you need to know in order to make critical decisions. It is important to share technical details that are necessary for making these decisions.

Documentation in the Hiring Decision

Another thing to ask about is documentation. Documentation is often the bane of everyone's existence around technical projects, but how a developer or systems administrator thinks about documentation can tell you a lot. Does that systems candidate keep network and server diagrams up-to-date and plan out changes visually before implementing them? Is the developer comfortable sketching out the pieces of an application he or she has built recently and explaining how the pieces fit together? And does that person understand that documentation isn't just about giving something to your boss or leaving notes for a co-worker but primarily about *writing down notes for your future self*? You might be rolling your eyes as you read this, thinking of all the undocumented stuff in your purview … and, sure, things haven't necessarily blown up around you because of the absence of good documentation. But on the other hand, if you have a candidate with acceptable skills and good documentation habits, you're probably better off with that person than with the person who sounds like he or she has better skills but can't draw a simple system diagram and explain its basic functions to a nontechnical person.

How to Keep Them Once You've Got Them

I'd like to leave you with a few more quick thoughts on another aspect of the question of how to hire good technical staff: how to keep them once you've got them. Good library systems people are always in demand, and even in a down market such as the one we're dealing with today, opportunities abound. Engineers and software developers often like a good challenge, so it can be difficult to keep them interested over the long term. And where most of us sometimes wonder if the grass is greener on the other side of the fence, a good engineer is looking around for a really high fence to design a climbing solution around. If you develop a good rapport with your technical folks, whether by using an approach such as painless functional specifications or any other, you'll come to know a bit about their strengths and the kinds of challenges they like. Some people like to learn a few tools really well and apply them to new projects; other people like to learn new things all the time while getting their work done. With this in mind, don't hesitate to lay down a difficult task once in a while. (By difficult here I mean something they'll think is difficult, which, hopefully, you'll trust them to tell you honestly if it is or not.) Sometimes a new system or service you think would help your community but fear might be too hard to build may be just the kind of project to get your geeks really going.

On the other hand, don't abuse the privilege. Most programmers I've worked with hate one thing above almost any other thing—to see their work go to waste. Everyone can understand over time how some projects just don't quite work out well. But nobody likes to invest time and energy on a substantial project only to see it thrown away, ignored, or forgotten for some higher priority system. And few technical people will tolerate that happening to them again and again. Just as they like a challenge, technical people like to be able to see their solution through to its use. It can be a great reward to see your stuff get used. So pick meaningful challenges, and pick your spots too.

IT'S ALWAYS GOOD TO HAVE A POSSE AROUND TO KNOW WHAT YOU'RE DOING AND TO BACK YOU UP.

Finally, if you are going to lose somebody, it can be easier to hire somebody who's good if you've had a good relationship with the departing person. As a programmer, if I visit a new environment and see people who communicate well, have a good sense for how to plan and manage a technical project, and have good documentation skills, and if the current staff is used to working with each other productively, it will all be obvious from the start, and it will be appealing. There are a lot of lone, conquering hero types out there, and if you have or can get one, that might work well for you. But in my experience there are far more people who like to work with others, not alone, to take on bigger challenges. For my money, it's always good to have a posse around to know what you're doing and to back you up, even if you're the main technical person. It'll be easier for you as the manager to keep things running, to bring new people aboard, and to designate responsibilities as people come and go if you've got a group of people who tend to work well and communicate well with each other. If you have that group, you're lucky; if you don't, you've got work to do. ■

Daniel Chudnov is a librarian working as an information technology specialist in the Office of Strategic Initiatives at the Library of Congress and a frequent speaker, writer, and consultant in the area of software and service innovation in libraries. Previously, he worked on the DSpace project at MIT Libraries and the jake metadata service at the Yale Medical Library. His email address is daniel.chudnov@gmail.com, and his blog is at http://onebiglibrary.net.