CHAPTER FIVE

Andy Lester on What Makes Developers Tick

We wanted to know what makes developers tick, and all signs pointed to Andy Lester. He's an expert in hiring smart people and building teams that work, and he's done a lot of writing and speaking about the topic. And as a member of the Perl development team, he knows what it takes to build great software. We talked to him to get some insight into what motivates—and demotivates!—the people who build software, and how to help them work well together.

Andrew: What makes you know about teams, how software teams work?

Andy: I've been programming for 21, 22 years professionally, on some good teams and some bad teams. First off, I program for fun. I'm able to do for a living and make good money in what I love to do anyway. If I'm not having fun doing this, it's not worthwhile. And part of that fun comes from people I'm working with. Yes, it's fun to work alone, but the social aspect—both at work and on open source projects—is undeniable. The open source world has social aspects as part of the whole thing, because otherwise, you're just writing for yourself.

Andrew: Do you think that's a big part of getting a team to work better? That need for social fulfillment is happening with everybody on the team, and not just some people?

Andy: That's hard to tell. You go back to Maslow's hierarchy of needs, and social is one of those things that's pretty low on the pyramid, meaning more basic. But what social means to some people is very different from what social means to others.

Jenny: Are you saying that in order to have a good team, you need to go out for pizza twice a week? What do you mean by "social"?

Andy: For some people, their "social" is sort of separate from the work to be done. So yes, it could be that we all hang out and have pizza, or we all go to lunch together. For some people, the social aspect is absolutely integrated into the work—"I want to work with other people." Some people want to throw on their headphones—"Don't bother me for the next four hours." Neither of those is right or wrong, they both just are.

Andrew: So you have to be sensitive to how the people on your team work, and help give them an environment they feel is the most interesting, most fun social environment you can give them?

Andy: Yes. And people coming into an environment need to be aware of what the existing team dynamic is. If I, Andy, have someone come to me and say, "Hey, I've got a problem figuring out what's wrong with this function here," I'm glad to help out, whereas other people may see that as a gross intrusion on their programming time. I see helping other people as much a part of my job as actually writing my own code. Not everyone sees it that way.

And it's tough, because we're so set in our ways that the initial reaction when those kinds of conflicts come up is to think that the other person is wrong. "Why are you bothering me? You are wrong to be bothering me." So I could see somebody else who's struggling on code as wrong to not avail himself of the human resources around him, and say, "Yeah, I could help you out on that." And understanding the dynamic you're getting into and already are in takes a lot of self-awareness and observing of the world around you.

Andrew: How did you learn that? Did you learn that through experiment? Did you have a bad experience?

Andy: I've certainly joined teams where I, the gregarious Andy who sees that sort of interaction as the way life should be, the way I prefer it, where that's not the case. Where

interrupting the flow is seen as a cardinal sin. It didn't go well. There was constant conflict because of it, and that was very frustrating to me because I felt alone and isolated, and I'm sure the other people felt that I was a pain in the ass.

Jenny: OK, but what about other people who, once they know that they can get help from people around them, tend to lean on other people all the time? People who run into the first roadblock and, rather than Google for the answer, tap someone on the shoulder every time?

Andy: Well, the way you phrase that tips your hand as to how you see that.

Jenny: Well, I'm asking, is it possible that it's negative?

Andy: It could be negative. It all depends on what the social norms are. One thing that geeks are terrible at is social norms. And I don't just mean social norms of society—"Hey, look, bathe everyday"—and I use that both humorously and with all seriousness.

What I'm talking about is social norms of any given group. And that can be really tough, because you have to have awareness of those around you. That lack of awareness can be debilitating, and actually I do have a story from yesterday, when we were having a code review. Three of us in the group were well seasoned with doing code reviews. The fourth guy, who was actually having the code reviewed, was less comfortable with them. We went in and were talking about the code, discussing it and so on, and as we kept going the guy getting the code reviewed was getting more and more uncomfortable with the things we were pointing out. And one of the other guys was becoming more and more animated, and more and more interested in explaining his point of view, and became oblivious to what the guy having his code reviewed was looking like. I was watching the reviewee, who was getting upset and frustrated at how things were going. The other two weren't paying any attention, they were pretty much just talking to each other, and that lack of awareness could have been really ugly. I saw what was going on, and said, "You know, guys, we've been at this for a while. Let's take a break." Just to let the pressure off.

Geeks—and I include myself in here—do not by nature gauge their audience well. And their audience is really anyone within earshot. In this case, there were four of us at the table, and all of a sudden it became a two-person conversation and there was no awareness of the other people in the room. If I hadn't stopped that, I think there would have been some real hostility popping up.

Andrew: So I'm the sort of geek you're talking about.

Andy: God bless you.

Andrew: And it's taken me a long time to realize this and change the way I interact with people. One of the things I'm in the habit of doing—and it was a lot worse when I was younger—is that when I've got a point, I'll be overly pedantic about it and argue it to completion, whether or not the other person is interested in arguing. It took me a long time to learn that I shouldn't make overly broad generalizations, and then let the

argument sort out the actual issue, because normal, non-geek people tend to find that extraordinarily irritating.

Now, I've had many geek friends over the years, many who've had similar problems and don't recognize it. In fact, they'd probably be surprised to find out that making generalizations and arguing them out is even a problem, and not the preferable way to interact with other people. What do you do when you've got somebody like that on your team? How do you help them? Or do you help them at all? Do you try to change the environment to suit them better?

Andy: A lot of it depends on what your relationship is with the offending person. For many geeks, conforming—that dirty word—could be seen as not only unnecessary but a sign of weakness. A lot of geeks pride themselves on being different. Difference is braggable. And that's great on its own I guess, if you want to be different. But if it hurts the team, then it's not.

Either people get it or they don't. Helping those who don't get it can be difficult, because that intrinsic need that all humans have to be understood, respected, and listened to, may be seen as a weakness by the geeks. That's a really tough battle.

Jenny: We're talking like it's just accepted that there's a certain kind of person who's in software. Is that true?

Andy: No, of course not. Not everybody is socially inept.

Jenny: Right.

Andy: But there is a not-insignificant percentage of people who are, who have grown up dealing with computers—preferring to deal with computers—because the computer does what you want, it deals with you exactly, logically. I don't have to worry about how the computer feels. I can get mad and curse at the computer, and the computer will not be upset at me for that. There are no annoying emotions to have to deal with, and dealing with other people's emotions can be annoying. Ask anybody who's been married or has had a long-term relationship. Yeah, it can be annoying. But it's what needs to happen if you're going to deal with other people.

Jenny: So how does this all come together? I'm interested in a couple of facets of what you just talked about. One is how people react to criticism in general. When you were talking about the code review example, it seemed like part of that was about people being aware of one another. But another part of it is that in a team environment, you have to put yourself out there: put your code out there, put your work out there, and allow other people to react to it, take that reaction and do something with it.

Andy: And one of the things is that you'll have somebody, for instance, who's able to separate himself—and I say "him" because it's overwhelmingly male—who will be able to logically separate himself from his code, his work. Somebody comes along and says, "Gee, you know, this code totally sucks here because you're doing blah, blah, blah." He's going to logically stand back and say, "OK, I never thought about it that way, I'll do it that way. " Whereas other geeks or, more likely, normal people, will not understand, will take that

as an attack, take that as an insult. "You did that wrong" is effectively what the person is saying. And that can be very harsh on the other person.

The computer guy will come around and see somebody in another department, for instance, and say, "You're doing your spreadsheet wrong, you should be doing blah, blah, blah." And he is working strictly on a logical point of view: "This work product could be better if this changed." Whereas the person whose work product it is that's not done according to the geek's standard sees it as an attack on the work that he's been doing, and that is something that's very difficult to understand. It's even more difficult to understand when it's a fellow geek who feels like that. That's the really surprising part. Because even if they learn, "OK, I don't go over to the people in accounting and criticize their spreadsheets because they do 'em wrong," they can understand that. But when it's another programmer? That's baffling! "Why does this person feel like that? That's crazy!" And not only that, because that would be almost empathetic, but, "Why are they so sensitive?"

Andrew: So how do you help a non-geek understand this? And how do you help a geek come around? How do you help them interact? It seems like we've got potential communication problems on both sides that really require a lot of understanding.

Andy: Number one, the geek has to understand that there is benefit to respecting others.

Andrew: It sounds so simple when you say it like that.

Andy: It is! It's funny because there just was a panel at OSCON with Kirrily Robert—she runs a website called GeekEtiquette.com—so Kirrily and I were both on this panel. Her section was an intro to etiquette, and why etiquette isn't about salad forks. I mean, it can be, but it doesn't have to be about how you do your wedding invitations. It's simply greasing the wheels of human interaction.

She said, using the 80-20 rule, the three things you have to remember about etiquette are respect other people, listen to what they say, and take a shower every day. If you can do that, that's your 80%. And the reason is that if you don't do that, people will not want to work with you or interact with you.

And the thing is, it's not that tough. Her whole point is that a term like *etiquette* is a challenge, per se—yes, sometimes it takes work to actually listen to what people are saying when they're annoying you. Because, yes, it can be annoying to have to listen to what other people say for the sake of respecting them.

The key is that the geek has to understand that there is value in respecting other people, and those social interactions are necessary. Because sometimes, it's a matter of "if you're a jerk, you're getting fired." You have to understand that if you want to do the things you want to do, that requires help from other people, and the only way that's going to happen is by being a person that people want to work with, want to do things with. And then, once you're there, then you can internalize the idea that while you may not take it as an insult to have somebody pick on your spreadsheet, many other people do. Many other people are not like you. And the differences between us can be so vast, between Bob over in accounting and the geek, the differences can be so vast such that you have to stand

back and say, "Wow, how can anybody think like that?" Well, you don't have to understand how or why. Just accept it and work with it. And respect that that's the way the person is.

Andrew: Wow. That's great advice. I wish I'd heard that a lot earlier in my life. But what about Bob in accounting? What's life like from his end, dealing with the geek? Can that have an impact on a team?

Andy: Bob in accounting has a tougher job, because Bob is more normal (and I mean statistically normal). You might have 100 people in the company, and only five of them are geeks, or jerks, or whatever. And he is going to feel very justified in feeling offended, because the social norm around him is "Yes, that's the way the geeks are, and you just have to kind of put up with it." But Bob would do well to also respect that the geek's way of looking at the world is also OK. The way he looks at it is that he's just trying to improve things for the sake of improving them, because that is what he does. He wants to improve technical things around him for the betterment of the person. Bob needs to understand that as well. He may not feel that way; he may not feel like the geek is trying to improve things, but rather tear him down. But Bob should know that that's all the geek has in his heart. Really, the geek just wants things to be better technically.

Andrew: Let's take it a step back. Now it's not just one-on-one, Bob and the geek. Let's take it to software teams now. Now you need to build up your team. Let's say you're on a team, and you need to help the rest of them. You start recognizing that you've got this problem. You could be the person in charge of the team, or you could just be someone who's an intrepid team member who realizes, "Wow, this really affects me and everyone around me." What's the next step? What do you do? How do you use this knowledge to help make your software better?

Andy: Well, you're not going to make the software better, per se, but you're certainly going to build it faster. Because short of a server catching fire, there's nothing more damaging to productivity than having people hate each other, or not being able to work together, or having to go to a manager and say, "Look, I need you to sort this out." As a manager, I dread that. It is the biggest time-suck in terms of the amount of effort I have to put into it.

I can deal with a server problem, and it's done. But if I've got people who are pissed off at each other, I may solve the problem that day, but it's not going to go away for a while. So certainly, it's in my best interest as the team leader or manager to not let that stuff happen. If there's any potential conflict between people, it needs to be dealt with right away.

Jenny: How much do you think conflict influences how your team behaves? By and large, how has conflict been an issue for you in the past in building software? Do you feel like that happens all the time, that two people on the team start to dislike each other and you end up having to have an intervention?

Andy: One big difference between Bob in accounting and the geek is how respect is seen. Society says that by default, you give a minimal amount of respect to everyone. It's ascribed. You say, "Hey, there's a person here, and I'm not going to be a jerk to this person

because it's just another person." To the geek, respect must be earned. And the lack of respect—if somebody does something stupid, in many geeks' minds that is worthy of disrespect.

I was on a team once where I said, "At the very least, can we just have minimal respect for everyone here?" And I was asked quite seriously by someone else, "Well, what if not everybody on this team is worthy of respect?" And that's baffling to me as a human, but it's also not uncommon. And that minimal amount of respect is something that many just don't get.

Jenny: It makes us sound like a really angry bunch.

Andy: Many people can be.

Andrew: But it also gives a kind of "secret back door" for a geek who wants to improve a working relationship with somebody. If he needs to work with Bob in accounting, and he knows that his paycheck depends on being able to work with Bob in accounting, show Bob a little respect—even if he hasn't earned it yet. Then he'll reciprocate and become a little easier to work with.

Andy: Yes. And the lack of respect—well, look at online culture. Why do you have flame wars? Because the person on the other end doesn't need my basic respect. I'm able to say the most disrespectful things, because the person on the other end (if you even think that far) is an actual person reading your words who is not worthy of your respect. That doesn't even have to be a conscious decision—"I'm not respecting this person." Respect doesn't even enter into it. The geek won't even think that far ahead.

Andrew: So let's say you're a team member, and you have to get your work done. You're reading this interview and you think, "Wow, that's me. That's everyone I work with. We are having trouble with Bob in accounting." What's an easy first step?

Andy: Here's the thing. We're talking about behavior here. If I have a problem with Joe on my team and he doesn't respect me, that's not a problem. The problem is the behavior behind it. If Joe is aggressive, antagonistic, that's the problem. You need to say, "Look, Joe. You and I need to work on this project, and for that to happen I need X, Y, and Z to happen. I need to be able to have a code review without hearing insults about my code." And everything is I, I, I—this is what I need. Not "Joe, you're a jerk because you're insulting my code and you don't respect me." You cannot make Joe respect you.

And if you point the blame at somebody and say, "You are doing this, you are doing that," now you're opening yourself up for argument. If you say, "Joe, you're disrespecting me"—"No, I'm not!" And you can have an argument all day, and sometimes that's exactly what happens, about whether Joe is disrespecting you or not.

But what you can say instead is "This is what I need. I need to have all discussion to be related to the code at hand. This is my base behavior." Because behavior you can't argue about. You can argue about attitudes or feelings or intentions. And if there's one thing about geeks that we know is that they love to argue, they love to debate. You don't want

to fall into that trap of opening it up for debate about whether Joe is being a jerk, disrespecting, etc. Anytime you're talking about that, you have to keep it strictly on the behavioral level.

The other thing is that this is something for your manager to deal with, or at least be made aware of. Really, your manager's job is to remove impediments to your work. If Joe is being antagonistic and disrespectful to you or anyone else on the team, that's an impediment to the work. What you're really doing is you're saying that this is an impediment to work, just as much as if my machine crashed and I didn't get any work done today. Both of them are detrimental to the productivity of the team.

Now if your boss doesn't understand that, if your boss is somebody who does not understand these sorts of team dynamics, that is probably a doomed team and you'd probably do best to go somewhere else.

Jenny: I want to take it up a step from geek etiquette and teams and talk more on a macro level about managing teams and working with people on teams.

Andy: A number 1, you as a team leader or manager have to understand that the team has to work together. Those people problems are your number one potential impediment, and anything that gets in the way of people working together is going to crush your project. You have to understand that you may have people on your team who are a net negative as far as productivity goes. If you have somebody who cranks out 100 lines of code a day but drags everyone else's productivity down, whether that's because nobody wants to work with them, whether that's because time that was previously spent coding is now time spent wasted in meetings trying to get everyone to get along, you have to understand that from a managerial point of view.

Given that, you have to realize that the people who are disrupting the team need to modify their behavior immediately. Otherwise, the project will go down the toilet. And if that person cannot understand that the team dynamic is more important than any individual contribution, then that person is the wrong person for your team and you need to help them out. And I mean, help them out of the team, out of the company, whatever it may be.

Andrew: What about when you're putting the team together? How do you avoid that in the first place?

Andy: If you're talking about hiring people to join your team, then you need to hire specifically for that. Typically, what I've done is that when I hire someone, I have at least two rounds of interviews. The first one is merely getting past me, and basic technical chops. Can you write code? What's your level of technical understanding of programming language X? Do you understand database theory? Whatever it may be. That's not tough. And it's not very time-consuming, either, because it's just me and that person. If that person passes that initial bar of "Can they write code? Can they do the work?" the second part is "Can they do the work with the team, and will they be a fit with the team?"

I say "fit" very specifically because every team is different. Every department is different, every organization is different. And as with anything else on these interpersonal issues, it's not a matter of right and wrong. There's nothing wrong with being the kind of guy who wants to go be in his cube and not talk to anybody for eight hours a day. There's nothing wrong with that, except that you are wrong for my team. If you want to work here, that's a bad choice. It's a bad choice for that person to work with a team that I'm on, because that's not the way my teams run.

So the second and sometimes third round of interviews is basically getting the rest of the team together and talking to the person. Technical issues will come up. But mostly, I'll sit back and watch the interactions with others, and see the kinds of things that they talk about. The rest of the team will ask things. Usually it'll be guided by senior developers. They'll come in and talk to the person and see how they expect to fit in with the team. That's when you talk about things like conflicts. "Tell me a story about a conflict you had with someone else on your group, when you thought the other person was pretty much full of crap. How did you deal with that?" Or, "Tell me about a time when you felt you were treated unfairly by someone on your team." "Tell me about a time when you've had somebody difficult on a team, and how you worked with that."

That's not the whole of the interview, but it's a big part of it. Just hearing a few of those stories—and it needs to be stories, not just "Do you get along with others on your team?" "Of course I do," everybody's going to say that—given that, you can really get a feel for what the person's going to be like in your group.

Andrew: So that second interview is not just with you, but with your team members, too?

Andy: Yes, that second interview is usually with the senior team members, and me, and maybe even my boss. There will be technical things discussed, but the real meat of what we're getting at is the interaction. What I tell people is that I'm really picky. I have one open position here, and I'm going to be really picky about who gets that position, because I love this team, I love the work I'm doing, and I'm not going to let just anyone join my group. And there are also, of course, more practical aspects to that. Number one, it's really a pain to fire someone.

Andrew: What about pitfalls for this approach? Does everybody get a veto? What if everybody likes the person, except one guy?

Andy: It's absolutely not a vote. It's certainly not a binary vote. It's "Tell me what you think." I'll ask, "Around the table, show me your thumb." But it's certainly not binary. And the fortunate thing is that I've never had that case, where it was clear that one person was out of alignment with what other people thought. I've never had a case where one person was saying, "Yeah, this guy's really great and we need to hire him," and everyone else thought he was a jerk, or vice versa. Because I find that when you talk long enough with people, it's pretty clear whether the person's a good fit or not.

And again, it's fit. "Are they going to really fit in with the rest of the team?" Now, the other pitfall that I do definitely worry about is that I don't want the team to become a

monoculture. I don't want it to be where we only hire people who are just like us, because then you only get people that are just like you, and you don't get a lot of innovation. I try to look for innovation in other areas.

Jenny: When you use words like fit *and* personality, *do people ever gravitate toward hiring someone they want to be friends with or hang out with, more than really putting them through the paces to see if they're actually qualified? What does that do to the culture of the team? Do you find that it affects the culture?*

Andy: I haven't found that, not that anybody would tell me that. Nobody would say, "I want to hire this guy because I think he'd be fun to go drink with." We have enough differences in what the team is about that not everybody's alike, so I'm not too worried about that. I think that everybody on the team who comes into the interview understands the importance of hiring someone they have to deal with every day. We're hiring someone who you'll spend more waking hours with than your spouse.

Andrew: Do you think that a good team should necessarily be made up of people who like hanging out together after work?

Andy: No, definitely not! I don't think that's a requirement at all. I've been on teams where there are people that I couldn't imagine going and hanging out with, because they're just not interesting to me in that way. Maybe they love to play beach volleyball, and I have no interest in that. We've got one guy here who goes and plays golf a couple of times a week as soon as work is over. Well, that's incredibly boring to me. But that's OK. By the same token, not everybody works on open source projects like I do. Not everybody plays guitar in a traditional Irish band, like Mike does; whatever it may be. Everybody has outside interests, and that's OK—they don't have to be the same. As long as we can get together for those eight to *n* hours a day and get stuff done.

Now that being said, sure, some people I do hang out with. And that's just natural. But it's not a failure if not everybody wants to go do it.

Andrew: So what about your open source projects? How does all of this apply to them?

Andy: The dynamics of people having to get along together on an open source project are much the same as the ones of trying to get along in a company. The difference is that on an open source project, there's nobody who can say, "Shape up and fly right or you're out on your ear."

Andrew: What do you think that does to the dynamic, that there's no one person on an open source team who can say that? Does that help or hinder?

Andy: Well, one of the things that's different on an open source project is that the leaders become leaders. Look at Perl, which I'm very involved in. Yes, Larry Wall is the leader of Perl, because it's his language, he wrote it and invented it. But aside from him, all other leadership has simply come from things that people have done. If you work on such and such, this part and that part, and you write these modules, and everybody likes your mod-

ules, and you help people out, well, then, you naturally are given respect and are seen as a leader. As opposed to "Larry has appointed so-and-so to be the boss of this group."

I think that really matches a lot of the geek tendency toward meritocracy, that the people who do the most and best work get the rewards. And if that reward is that the person is seen as a respected member of the community, then the two naturally go hand in hand.