From: www.itworld.com

Mentoring in Open Source Communities: What Works? What Doesn't?

by Esther Schindler

September 20, 2009 —Some software developers wrinkle their nose at the very idea of deliberate one-on-one help. The default behavior in many (most? it's hard to know) free and open source software (FOSS) communities is to read the code and documentation, try it out, rinse and repeat. People improve their skills on their own; if they need help, they post a note in a forum or ask in IRC. Why should it be otherwise?

However, we all learn differently. You might want to settle in with a programming book, while I prefer to take an in-person class. If your project wants to attract new contributors, it behooves you to think past the "dive into the deep end" culture.

"Many people want to participate, but are literally sitting there just waiting to be asked," says Mel Chua, a new community engineer on Red Hat's Community Architecture team and frequent contributor to the Sugar Labs and One Laptop Per Child projects. "No amount of general 'Just jump in, please join us!' shouting will work if the potential contributors you're trying to reach don't think that any such invitation could possibly be directed towards them. If you want new contributors, you need to go out and find people and *ask* them, individually, for their help," she says.

Many open source communities do actual mentoring (even if they don't think of it with that label); others don't. Some projects make a concerted effort to connect newbies with more experienced people. They provide opportunities for people to work together in smaller teams (not just a gang hanging out in an IRC channel, however useful that is), such as in sprints and code-a-thons. The best known effort is undeniably the Google Summer of Code (GSOC), but plenty of mentoring happens on a smaller scale.

I asked people from several open source communities to share their mentoring experiences. In this article you'll learn what works, what doesn't, and how to encourage these relationships.

> *In talking here about mentoring, I make a distinction between "learn on your own" (such as examining the changes others make to the code you contributed) and somebody offering specific, individual advice (e.g. "It might run faster if you did this..."), particularly in an ongoing personal relationship. For more on mentoring in general, see my older article, The Executive Woman's Guide to Mentoring.*

Just like any family, company or community, open source communities have their own way of doing things, says Grant Ingersoll, a member of the technical staff at Lucid Imagination, the commercial entity to Apache Lucene and Apache Solr enterprise search technologies. "If the project is going to be sustainable, you have to be welcoming of new members." And it's important outside the community, too; sometimes mentoring involves helping companies who are hesitant to adopt open source feel more comfortable, Ingersoll says.

Nor does the effort help your project attract only the raw, wet-behind-the-ears new programmers. It builds loyalty and commitment. For the last ten years, Chris Cormack currently the translation manager for the Koha Open Source Integrated Library System, has acted in a mentor capacity to several people. "I think almost all the people I have spent time with are still active in the community," he says. "Or [they] made a positive contribution to the community/project before they went on to other things."

Creating Mentoring Relationships

It's unlikely that you could be involved in the open source community and remain unaware of Google's yearly sponsorship of open source projects. "Since its inception in 2005, [Google Summer of Code] has brought together nearly 2,500 successful student participants and 2,500 mentors from 98 countries worldwide, all for the love of code," summarizes its website. The company has collected plenty of hard-earned wisdom, and I'll impart some of the lessons learned by Google's Leslie Hawthorn when we get to the specific advice section. (Really, it's coming.)

However, Google is not the only company to contribute in this way. The Ubuntu community, for instance, uses mentoring extensively. Explains Jono Bacon, Ubuntu community manager, people who want to become Ubuntu developers engage in a sponsorship process. The developer contributes a package, which existing developers review; the experienced developers offer comments until the prospective developer is ready to directly contribute to Ubuntu. "We have also run extensive mentoring campaigns in other parts of the community, helping teams to mentor other teams, mentoring at a governance level and more," says Bacon.

Deliberate mentoring is not limited to the "household name" FOSS projects. Abizer Rasheed, president of Effortless 24x7 and sponsor of SimWitty (which I also referenced in my article about sponsoring open source sprints) conceived of SimWitty's mentorship model as a three-way partnership. A sponsor provides high-level guidance, an advisor does the day-to-day mentoring, and then there's the intern or mentee. "Our pilot has my firm being the sponsor, J. Wolfgang Goerlich (a local seasoned security expert) mentoring, and a college student from Detroit interning. We plan to scale the model up with additional sponsors, advisors, and interns," he explains.

While it's ideal to have someone else create the connections, however, it isn't at all required.

Says Apache Lucerne's Ingersoll, "The deliberate ones (GSOC and a few others) involved explicitly recruiting certain people based on their credentials, etc. The others likely just start from exchanging e-mails, answering questions, helping them work through their contributions."

Plenty of mentoring relationships evolve spontaneously. Joe 'Zonker' Brockmeier, openSUSE community manager at Novell, has seen people interested in packaging for openSUSE who started by asking a few questions of the maintainers. As they continued working, it evolved into a more consistent mentoring relationship, until the packager became competent and started running on his own.

One delicious example of a spontaneous relationship that worked is the one between Koha's Cormack and Nicole Engard (http://web2learning.net), Koha's documentation manager and soon to start a new day job as director of open source education at ByWater Solutions. Engard joined the Koha community in 2008. "I was excited to be a part of something so awesome -- and rare in the library world -- but a bit nervous about jumping into something that was so well established," she says. Cormack, one of Koha's original developers, took Engard under his wing. "He taught me the ropes, he never, ever said I had a stupid question. He never ignored me. He has always been patient and willing to walk me through things." Distance didn't matter; Cormack is in New Zealand, and Engard is on the U.S. east coast. "There is a huge time difference for us; he is often helping at midnight his time," she says with heartfelt appreciation.

That's not to imply that every would-be contributor appreciates help. Some folks never quite grok that an expert's desire to "pay it forward" means the expert owes you that favor to you personally. Cormack's mentoring relationships "have all evolved from people asking questions on IRC or in the mailing lists. Then following up the answer with a thank you; usually the 'thank you' is what gets me hooked."

A Culture to Encourage Helpfulness

One major influence is creating a welcoming environment where people feel safe enough to "ask a dumb question." How okay is it, in your community, to have no idea where to start? How clear is that path for those people? Who steps forward to make them welcome?

Why yes, I *am* thinking about that grouchy person in IRC who answers all nervous tentative questions with "RTFM." Not everyone needs to be the Welcome Wagon -- but someone ought to be.

My definition of mentoring encompasses one-on-one relationships but groups can have that role too. A computer user group is inherently about educating others, especially when it actively encourages new members to get involved. That was so for Jim Keenan, founder of Perl Seminar New York, who got involved in the Phalanx project (it aims to extend the test coverage and documentation of important CPAN distributions) organized by

Perl Buzz's Andy Lester. "Three or four of us met once a month in a bar on a Saturday afternoon for about four months. Only one of us was a full-time IT guy; it was the first time I had worked next to someone employed at a fairly high level in the business."

Cross-team partnerships are another way to promote mentoring, according to Denise Paolucci, co-owner of Dreamwidth Studios, a LiveJournal fork. "We have teams for things like site documentation, technical support, QA/testing, accessibility, styles (journal layouts), etc.," she says. Those teams commonly work with each other and with the programming/development community. "If someone's working on a new feature, they can post and say 'Hey, I'm doing this thing and I need an opinion from someone in accessibility about whether I should do this or that.'"

Sometimes, participants have to earn the attention of senior contributors. Frank Koehl, founder of Fwd:Vault, cited his experience with the Zen Cart project. It fosters a vibrant support community, he says; however direct, unscheduled contact with team members is strictly prohibited. Team attention came to Koehl after he released a custom reporting tool and several other modules to the project, culminating in an offer to join as a support team member. "Here, (finally!) begins the mentoring," he says. "The team shared their private development plans so I could coordinate my modules with the release schedule, and offered direct advice on how to improve my offerings. In turn, I provided feedback on my experience with the program, offering recommendations on future areas of improvement."

Enough already with the heartwarming stories. It's time to get to specific suggestions.

Mentoring is About Communication

Let's start out with a few ground rules, summarized well by Chad Outten, director of Learning Technologies at My Learning Space. He initially volunteered to document support material for the Moodle teacher certification (MTC), completed the certification himself and was asked by the program manager to mentor-assess future candidates. In his opinion, MTC's mentoring is a key factor to successful learning outcome for candidates. His overall guidelines are:

1. Establish clear expectations
2. Communicate regularly
3. Provide structure through milestones and deadlines
4. Encourage constantly
5. Model best practice

Probably the most important of those is: Communicate regularly.

Google's Summer of Code program manager, Leslie Hawthorn, likely has more experience than does anyone else in what does and doesn't work in open source mentoring. She pointed me at a few outstanding resources, including Advice for GSoC Mentors and Advice for GSoC Students, which could be applied to nearly any project. After a mentor summit at the end of October, she says, they'll put together their guide to mentoring; it'll be available first from FLOSS manuals and later in a print edition.

What doesn't work? "When both sides assume that the other will do the work to maintain the relationship," says Hawthorn. Mentors should be approachable, not only at moments of crisis or when the mentee reaches an impasse. Her advice is for participants to decide how much time should be spent trying to figure out the answer: an "n hour" or "n minute" rule. That is, "If you get stuck and can't figure it out in an hour, e-mail me." As Hawthorn explains, "You don't want to train them to be so independent that they don't ask for help when they need it."

It's not unusual for mentors to wrongly decide that "no news is good news." Good mentors take time to strengthen the social bond, she says, and engage with the mentee early. Don't assume that a mentee will come to you. Offer advice like, "I thought you might have a little trouble with...." Hawthorn explains, "You uncover places that the mentee needed help with; it opens up a dialog."

Engard's Koha experience is a good microcosm of mentoring-done-right. "In the beginning, [Chris] just taught me about the basics of using the system, how to participate in the community, what software I should install to be able to communicate best with my new colleagues/community members across the world. As time went on, he taught me how to add code to the codebase and even explained the file structure so that I could find the right files to alter. Now he's helping me learn Perl."

Mentors should recognize that newbies aren't always confident about what to ask. Engard says, "If I'd change anything it would be to ask more questions. In the beginning it can be so scary to jump into a world that has been well established. There is a culture and pecking order that isn't always obvious, and I would have hated stepping on someone's toes. Now I know that everyone wants to help, they want new members in the community and [I] understand that the community is the power that drives the software."

According to Hawthorn, successful mentors are willing to share their own failures. It gives confidence, she explains. "If the god of the codebase you're working on admits that they made the most colossal blunder, their own mistakes don't seem so traumatic or horrifying." Similarly, an effective mentor knows when to ask someone to help out; most mentors explain, "I'm responsible for you, but you can come to any of us on this list." It also demonstrates that the "expert" mentor has plenty more to learn.

This ties into something Dreamwidths' Paolucci says: "Create a project culture where trying new things -- and failing at them on the first try -- is not only culturally acceptable, but viewed as a good thing." In many open source projects, she explains, imperfect or unfinished patches are viewed as a time-wasting liability, even if it's only in the minds of the people who are doing code review. "Even if you never *say* something like that where anyone can see it, the attitude is still going to creep into the subtext of every interaction, and newcomers will be able to sense it," says Paolucci. To attract new contributors, she says, it's critical to "create a culture where a partway-there patch isn't viewed as 'Great, now we're going to have to spend the time to clean that up' but as 'Great! Someone else did the first 70% of this, which is going to save me a lot of time and effort!'"

Mentoring takes time

Nearly everyone I spoke with emphasized how much time and effort it takes to properly mentor other people. It's worth it, they insist, but only if you allocate time for it, and only if the right people get involved. Helping the new people learn the ropes does make your own work take longer, Koha's Engard acknowledges. "But in the end, it means less work for you because someone new is on board to help out."

"The main problem with mentoring in a community project (one without significant paid staff) is time," says Josh Berkus, who's on the PostgreSQL core team. "Being a good mentor requires a lot of time, frequently more time than it would take you to do the development yourself. For this reason, most OSS developers are not temperamentally suited to be mentors."

Don't underestimate the time it takes to effectively mentor. "If I could reset the clock," says Novell's Brockmeier, "I'd probably structure my duties so that at least 25% of the time was set strictly to enabling new contributors." He recommends that companies with contributor-facing positions consider devoting at least 20% of their time strictly to working with new contributors -- one day a week with no other requirements. Even in a personal relationship, adds Google's Hawthorn, "Five hours a week doesn't sound like much but finding that time is difficult."

Don't expect immediate results. Says Dreamwidth's Paolucci, "It took us about four months of heavy, active development -- 100+ commits a week -- before we started seeing the payoff. You will lose people you mentor; you will spend incredible amounts of time and effort on people who don't stick around; you will lose the hack time of your primary mentor for at the very least a few months as it gets directed into reviewing, committing, and mentoring instead." She stresses: This is a long-term strategy, not a short-term one.

Don't forget to take notes! A mentor has a unique opportunity to learn a newcomer's perspective. Koha's Cormack wishes that he documented more, because "a lot of the questions are indeed FAQ ... so I do end up answering the same thing." That can lead to a more terse answer than the person asking deserves. If you take the time to write a thorough and complete answer to your mentee's question, consider also adding the answer to the project wiki; it can make it easier for the next person to learn.

The best FOSS mentoring experiences, it seems, comes from projects that devote attention to creating an easy on-ramp for new users, often including a senior project role whose job it is (formally or otherwise) to connect people. Let's take a closer look at that.

Put Someone in Charge of Mentoring

As PostgreSQL's Berkus said earlier, many developers are not suited to social interactions. That's fine; we all contribute based on our strengths and our interest. But several FOSS experts argue that projects ought to have someone who supervises mentors, connects people, or otherwise takes action on community health. At least one person on your project should have excellent social and interpersonal skills, and it's a skill that needs to be

appreciated. "A lot of geeks I know deride that kind of thing as touchy-feely bullshit," says Dreamwidth's Paolucci, "But it *does* lead to a much smoother community experience and a far more comfortable environment."

Berkus points out that the person supervising the mentors might be a member of your project core team. "Or it could just be a committed, diplomatic, and fair person with lots of free time who cares about new people learning the project." (I hope this makes you think, "Hey, I know *just* the right person we should ask...")

It's also pragmatic. Says Ubuntu's Bacon, "While you can have a general mentoring campaign, it is always more effective to directly connect the dots and hook a mentor up with a student." But there are other relevant duties. One example is helping up-and-coming leaders and prominent contributors to learn from established leaders, says Bacon. "I have also helped connect Ubuntu LoCo teams directly to each other, and members of my team have helped build mentoring relationships around developer education and bug workflow," he says.

For a successful Summer of Code, for example, the SoC administrator is probably more important than the SoC mentors, according to Berkus. "PostgreSQL didn't participate in SoC this year because we couldn't recruit a good administrator," he says, citing as the responsibilities:

- Establishing the mentoring project's standards and deadlines.
- Performing status checks on how the project/mentoring is going.
- Arbitrating arguments between mentor and mentee.
- Getting a substitute mentor if the first mentor doesn't work out.
- Arranging recommendations/credit if the mentee needs them.
- Making sure that the mentoring project isn't wasted by making sure it's still in line with project goals/deadlines.

A primary role for Nancy Garrity, Alfresco community manager, is making it as easy as possible for the community to contribute. "Our community is growing fast; we now have more than 100,000 members so there are always new people who need help." The community manager role for Alfresco extends beyond mentoring, such as forum management, code camps and local user groups.

Measuring Success

For the Google Summer of Code, the success is measured purely on what the mentors think of their students' progress. Hawthorn says, "Success is going to be dependent on what you ask them to do, and that depends on who the person is. Sometimes your biggest victory is to get the person to communicate more openly." For example, the mentee might have become a subject matter expert on a particular codebase, and at the end of the exercise she's more willing to answer questions in IRC.

Every project defines its own way to judge progress. For SimWitty's Rasheed, it has little to do with developing code and more to do with developing opportunities. "Considering what's going on today in the market, in Michigan and Detroit in particular, there is a real need for help. People are looking for employment. Employers are looking for technical skills. But there is a gap between what people know and what employers need. So SimWitty is a way to grow people, get them experience, and eventually connect them with employers. SimWitty allows us to create a win-win-win model."

It is, undeniably, a rewarding experience for both mentors and mentees. Lucid Imagination's Ingersoll has formally mentored two students through GSOC and done informal mentoring for various Apache Software Foundation projects he's involved in. Some of his personal gains were measurable, such as increasing his knowledge of machine learning (the mentee knew more about the implementation). "I think you learn to be more patient ... Besides, it's a way of giving back for all that I've gotten from community," he says. "I also try to encourage new programmers to pick an open source project to contribute to early in their career, as it will often open doors for them."

The benefits are not only warm-and-fuzzy; they can also be good for business. The relationships Koehl made in the Zen Cart project extended well outside its borders, he says, and today he considers the team members professional colleagues. "I contact the team members any time I needed advice or support on my projects, Zen Cart-related or not. I just had a phone call with one of them last week to discuss online billing setups for my current project, Fwd:Vault. Who better to seek help with online payments than the programmer who's checkout code has reached "featured cart" status with PayPal?"

The mentee, of course, gains technical knowledge -- and often a lot more. Mel Chua, a new community engineer on Red Hat's Community Architecture team, says, "My open source community mentors have literally changed my life. They've taught me skills, introduced me to people, and even sent me on trips to places I'd never be able to visit otherwise. But more importantly, they've taught me different ways of thinking and looking at the world. I'm starting my dream job at Red Hat this coming Monday because of them; I consider myself the luckiest person in the world because I get to join the same team as several of my mentors and essentially learn from them full-time. It's an honor to be able to apprentice under people like Max Spevack, Greg DeKoenigsberg, Karsten Wade, and Paul Frields."

Ubuntu's Bacon sums up the benefits when he urges open source projects to make use of mentoring. "It is a powerful method of sharing skills and ideas, but start simple. Define some relationships and check in to see how things are progressing. Mentoring is not about hooking two people together and then forgetting about it. Check in with them to see how they are doing, how they are progressing and what help they may need."

Want more open source analysis and advice? Sign up for an RSS feed for Esther Schindler's Great Wide Open blog, and sign up for the IT World open source newsletter. You might even want to follow Esther on Twitter.

ITworld