Fedora Project

Karsten Wade
Community Gardener
Red Hat Community Architecture Team

ideas

ideas

The Key Questions
for community investment

1. Does this project have a chance to **transform** Red Hat's business?
2. Is this investment **low cost / high value**?
3. Do people **outside** of RH care **passionately** about this idea?
4. Are there clear ways to invest in **community capacity**?
5. Is there a **simple plan** to get started, with **aggressive goals**?

The Key Questions
for community management

1. Is the project **progressing** towards its goals?
2. Do contributors feel like their work is **meaningful**?
3. Is community participation **growing** over time?
4. Is strong **leadership** developing within the community?

Fedora Project

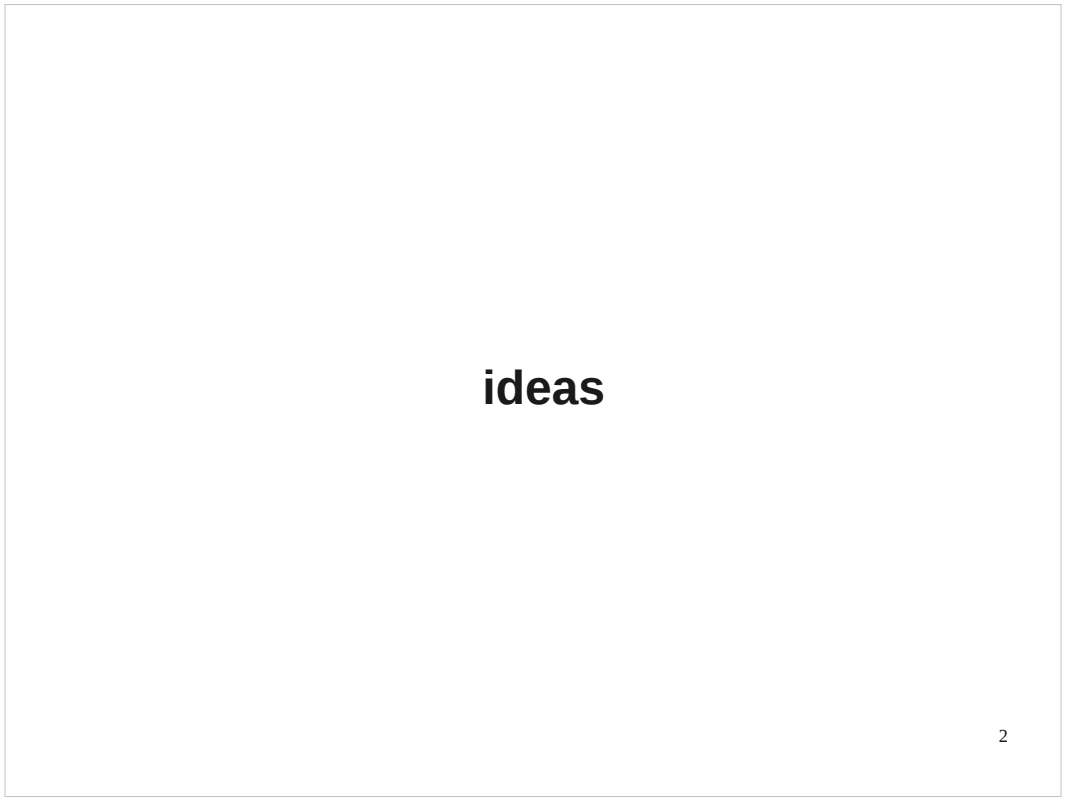
Karsten Wade
Community Gardener
Red Hat Community Architecture Team

Not here to talk with you about Fedora or why you should choose this distro over that. This last weekend, my man Zonker Brockmeier, OpenSuse Community Manager, told me that when he is asked at tech conferences why someone should choose his distro, he says, "Look, I've got a favorite brand of underwear, and it's my favorite because I've tried on many different brands and styles. You have to try on different distros to see what kind of underwear you prefer." I agree with Joe, I don't want to be telling you what kind of underwear to choose. If you want to talk about why I wear what I wear, ask me at any time. I also intend the substance of what I say today to be representative of what we do in Fedora and why we think we do it the right way.

We're not really talking about altruism today. People code free software for very, very selfish reasons. We call that scratching an itch. Would you call it altruistic to scratch the itch of your sister? Your mother? Your best friend? Probably not, especially since you expect a scratching back, someday. Would you offer or accept a backscratch from a stranger? Perhaps, but less likely, and most likely not every single day ... unless that is your kick.

Everyone who works on open source software does it for selfish reasons. From DJ Delorie writing a solitaire for his wife who preferred the one that shipped with Windows 98, the many, many contributors to the Linux kernel, working on sub-systems for their employers or because they want their webcams to work.

Why might you work on open source software? What is going to compell you to file a bug report, create or update a wiki page, own a package, commit code to an upstream? Look to your passions to know, that is what is going to keep you up at night working on a problem. Because it is truly important to you.



At the core of collaboration is an open sharing and building on ideas. Ideas introduced openly and under an agreement that they can be shared, reused, and built upon forever are ideas that can take on a real life. One person, no matter how much genius they have in their minds and fingers, can encompass as many good -- and bad -- ideas as a large project of people. One person, or a small team, can code up a nice application that solves real problems. But they are always going to be limited compared to the millions of people open source brings to bear on problems. You just cannot hire that many people.

Open collaboration works best when you iterate through as many ideas as possible. Humans are particularly good at generating ideas as part of a group, and in most cases, the result of that work is going to be better than one or a few people. It's a simple matter of memetic evolution. Ideas fight for supremacy.

When you get yourself involved in a community like that, one that cares about your ideas and nurturing them in to existence, it is a lift of warm air under your wings.

What I mainly want to talk about is how you can bring this activity in to your classroom and beyond.

Too often, the work we do in schools is throwaway. I have a shed at my house with multiple boxes stacked high with schoolwork from over the years. I've already recycled everything from high school, so this is all my writing and essays and poems from university. At the moment it is meaningful only if I drag it out, type it in to a computer, and release it under an open license. Otherwise, it is just boxes of dead tree.

If you are a computer science student, your classwork is peppered with programming that might just be a repeat of what was done by last years class. You learn some patterns and styles, but the actual output of your work is even worse than what is in my boxes.

Why not do that same work, same learning, but with an active, living codebase?

You are then learning from something that is both education-world and real-world, possibly on something more meaningful than just a programming assignment, and you are doing it in a way that builds your personal brand.

ideas

We talk about brand alot in business, and personal brand is what you build up over the years. It is the sum of your work experience, how your employers and colleagues feel about you, and what shows up when someone googles your name.

By starting right now working in open source software you are:

* Learning in an open environment where your mistakes are welcome; remember, bad ideas are as important as good ones.

* The stakes are lower than if it were an employment situation, the forgiving better, and the mentoring is extremely valuable. Some people will be unkind, some too kind, but all are looking out for your participation in the project. They are not trying to undermine you over the next promotion. Their mentoring is designed for the health of the project, which requires healthy and happy contributors.

* Building your personal brand and work experience years ahead of when it usually occurs. At Red Hat, some of our best coders started working as interns in college, then were hired full-time on graduation. They got the internship on the strength of their already proven abilities in the open communities.

* If you don't like an open source project, you can leave for a new and better opportunity. You don't have to quit and get hired somewhere else. You just sign off there and sign on here. You can work on projects that are in 'competition', because no one really cares the way a business does. There really is no such thing as a conflict of interest when the interest is in better software.

Thus ... if someone is offering a class that has working on open source projects as part of the coursework, flock to that class. If your instructors are having you work on dead tree assignments instead of a living branch of a real codebase, tell them what you really want to work on. Find the projects you care about and get involved today. Don't do this because it is helpful to your future. That part will take care of itself if you do your work now from your passions and ideals.

The Key Questions
for community investment

1. Does this project have a chance to **transform** Red Hat's business?
2. Is this investment **low cost / high value**?
3. Do people **outside** of RH care **passionately** about this idea?
4. Are there clear ways to invest in **community capacity**?
5. Is there a **simple plan** to get started, with **aggressive goals**?

Let me take a moment before talking about why it will be compelling to you, to focus on how we build community in Fedora. The fact is, Fedora is a community of people who output many, many things, one of which is a Linux distro. Why do people do that and why does it make better software?

When we are looking at where to invest energy, time, and money in the open source community, there are some specific questions we ask:

1. Does it have the potential to transform our communtiy and business?

2. Is the investment low cost/high value?

3. Do people in the real world care passionately about the idea?

4. Are there clear ways to invest in community capacity?

5. Is there a simple plan to get started, with aggressive goals?

What is transformative? The answer there is a blend of instict and intellect, mixed with experience and enthusiasm.

Low cost/high value -- take my Docs example. It used to be low/low, which is OK. Right now it is high/high, which is good. Soon I'll move it to low/high, which is great.

Our experience shows that people need to care about the idea, people who are going to contribute, participate, and consume as customers and users.

Community capacity is about making sure you can grow the community without stifling it.

Simple plans get off the ground faster, are less likely to get bogged down, and incite fire in people.

The Key Questions
for community management

1. Is the project **progressing** towards its goals?
2. Do contributors feel like their work is **meaningful**?
3. Is community participation **growing** over time?
4. Is strong **leadership** developing within the community?

How do we manage these investments?

1. Is the project progressing toward its goals?

2. Do contributors feel their work is meaningful?

3. Is community participation growing over time?

4. Is strong leadership developing from within the community?

Progress toward goals is key, as well as the flexibility to adjust the goals for good reason. Speed doesn't matter, the speed is inherent to the needs of the project and community.

Contributors need to see their work matters, to be thanked, to get credit, and so on. This is back scratching territory.

Community growth is not about endless growth, it is more like a sustainable garden. It should not grow and choke itself, nor be starved for resources.

Leaders arise from the group, not by being imposed upon them. Our formula here is simple -- get in, get movement, get out of the way.