What's In It For Me?

How Your Company Can Benefit from Open Sourcing Code

Brian W. Fitzpatrick
Chicago Engineering Manager
Google

ApacheCon, November 14th, 2007

Overview

- These are my opinions
- Target audience: corporations and folks in them
- Different strategies for open sourcing code
- Pros and cons of each
- Prescribe best practices

Why Go Open Source?

- Some sort of net gain
- Create better software
- Create a real relationship with your users
- Choose your goals
  - PR?
  - Goodwill from techies?
  - Free labor?
  - Change the industry, take over world?

Measuring “Health” of Open Source

- Lots of usage (not users!)
- A number of active developers
- Constant improvements and releases
- Remember: no community == dead software

Open Source Strategies

0. “Fake Open Source” Approach

- “Open Source” your code, but don’t use an OSI-approved license
- Pros:
  - PR splash
  - Effortless
- Cons:
  - You’re not open source
  - You’re missing the benefits
  - Open Source zealots may burn your house down

1. “Throw Code Over Wall” Approach

- Post tarball of the code, then walk away.
- Pros:
  - PR splash (maybe)
  - Effortless
- Cons:
  - No community to keep software alive (bit-rot)
  - Real techies give little cred

2. Develop Internally, Post Externally

- In-house development, public repository
- Pros:
  - PR splash
  - Occasional volunteers can send patches
  - A bit of cred from real techies
- Cons:
  - Community & momentum is wholly internal
  - External community likely to form elsewhere
  - Attracts only “follower” developers. (No bus keys!)
  - General distrust: only care about corporate agenda

3. Open Monarchy

- Public discussion, public repository
- Committers are mostly employees, occasionally a volunteer is given the keys
- One entity (corporation, lead developer) “rules” project and makes all decisions

3. Open Monarchy

- Pros:
  - PR splash
  - Even more cred from techies
  - Better quality volunteers; they can participate in discussions, sometimes commit directly
  - Results in better software
- Cons:
  - Community not long-term sustainable
  - High risk of angry revolutions and forking
  - General distrust: only care about corporate agenda

4. Consensus-Based Development

- Almost everything is public
- Decisions are based on consensus of the committers
- Commit privilege must be *earned* by everyone

4. Consensus-Based Development

- Pros:
  - Continuously increasing PR benefits
  - Long-term, sustainable communities
  - Complete techie cred
  - High quality volunteers (full bus keys)
  - Trust from other companies and participants
  - Results in *even better* software
- Cons:
  - Little short-term benefit
  - In the short-term, project agenda must come first
  - Hard Work
  - You need to hire strong leaders

Why We Think This Is Best

- Traditionally companies isolate developers from users
  - "They can be more productive"
- Results in better software
- If done right, internal developers will see the benefits

BUT BUT...
I Don’t Want To Lose Control!

- *“Strangers will force me to do things!”*
- *“Nasty people will hijack the project!”*

Answer: Craft your Community

- Choose a well-scoped mission
- Have your devs establish a strong, respectful culture
- Set the discussion tone carefully
- Have a well-defined process for making decisions
- Watch our ‘poisonous people’ talk ;-)

- Remember, you can set the stage, but it takes effort

What about Forking?

- Extremely rare in consensus-based development
- Majority always moves in one direction
- Really hard for a hive to swarm without at least 50% of the bees

How To Build a Consensus-Based Open Source Project

1. Come up with a Goal.

- Something useful
- Something people can be excited about
- Might only benefit your company indirectly, or in the long-term

- Examples: Collabnet, Google.

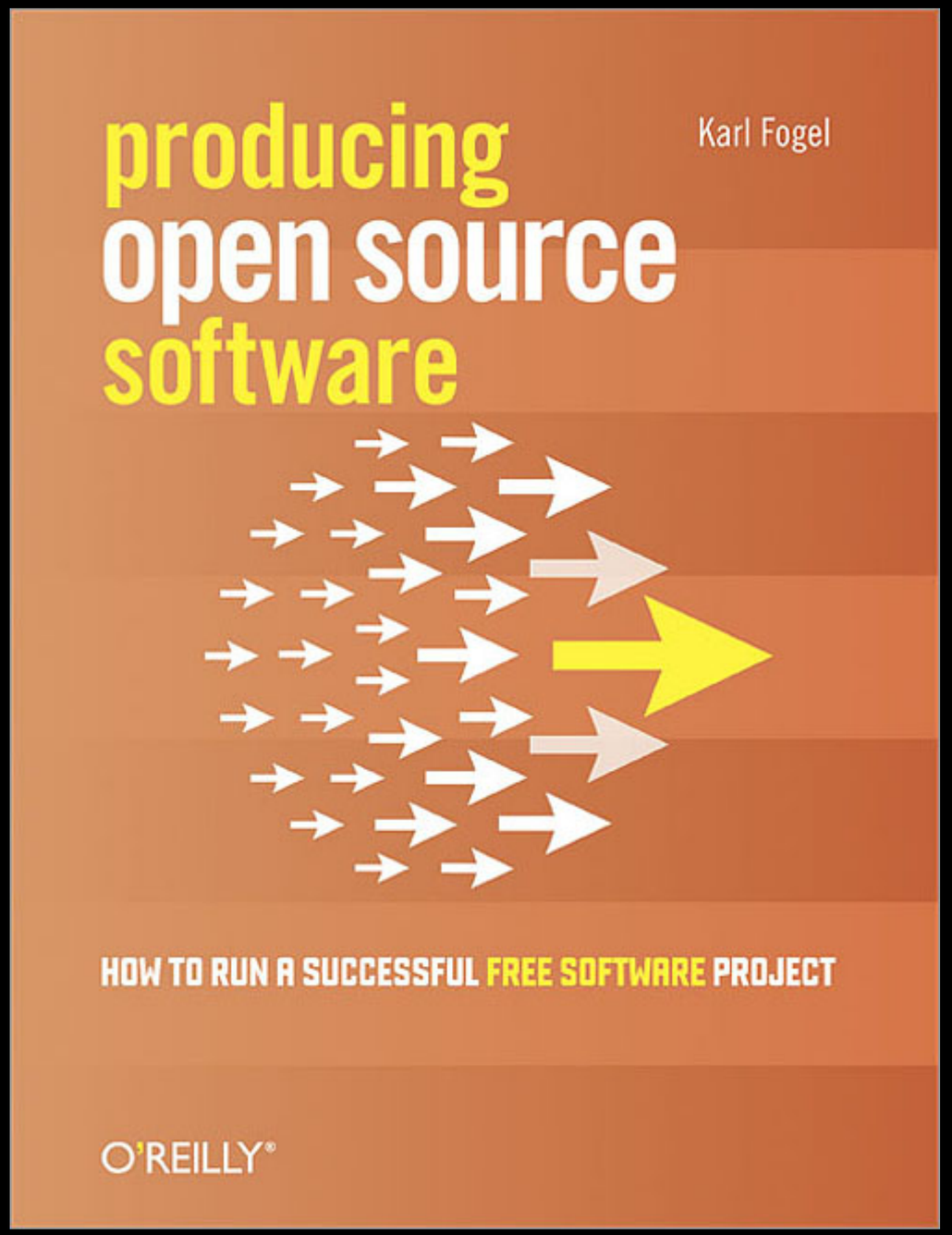
2. Write a Mission Statement

- Be very careful about scoping
  - too broad: attracts the wrong contributors
  - too narrow: attracts no interest at all
- Non-goals are important

- Examples: Subversion, GWT

3. Prepare your Team

- Read Karl's book! Discuss it
- Learn how to set community tone
- Decide on process for admitting new committers
- Learn how to diffuse poisonous people
- Thicken everyone's skin



4. Move all Development to Public

- Launch public mailing lists, repository, bug tracker
- Minimize use of internal mailing lists!
  - Develop policy for working with internal devs
- Do some PR to attract volunteers
- Start with one mailing list if possible, split later

Summary

- Choose strategy based on your goals
- There are tradeoffs
- We think consensus-based creates the best software

Q&A

Brian W. Fitzpatrick
fitz@google.com