So you wanna be a Linus Torvalds?

Désilets, Alain

Publisher's version / Version de l'éditeur:

Extended Abstract at the Conference on Engaging in Open Source 2006 (CEOS'06) [Proceedings], 2006

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*So you Wanna be a Linus Torvalds? The Do's and Don't of Running a Small Source Project **

Désilets, A.
June 2006

* published as an Extended Abstract at the Conference on Engaging in Open Source 2006 (CEOS'06). Halifax, Nova Scotia, Canada. June 1-2, 2006. NRC 48551. (Invited Talk).

Canada

So you wanna be a Linus Torvalds? The Do's and Don't of Running a Small Open Source Project

Alain Désilets
National Research Council of Canada
alain.desilets@nrc-cnrc.gc.ca

Introduction

In this paper, I present a series of lessons learned on how to effectively run a small Open Source experience. These lessons are based on my own experience as the leader of VoiceCode [2] [6], a project that aims at developing an integrated programming-by-voice toolbox. By this we mean tools that allow programmers with Repetitive Strain Injury (RSI) to write computer code by talking to their machine instead of typing.

The VoiceCode project started in 1999 by the National Research Council of Canada, and was first officially released in 2003. The system is now at a point where it can be used by programmers to do real work, and there have been over 7200 downloads so far. The project has also attracted the attention of the media [4].

In the process of leading this project, I have learned many important lessons; too many to discuss exhaustively here. I will however share three that seem particularly important.

The elusive “plausible promise”

When I started the VoiceCode project, I naively expected that I could get away with developing only a small core part of the system. Just enough to get people excited and foster the emergence of a community of dedicated contributors that would extend it into something usable. After all, this approach worked for Linus Torvalds, and it is what Eric Raymond advocates under the term “plausible promise” [5].

> *“When you start community-building, what you need to be able to present is a **plausible promise**. Your program doesn't have to work particularly well. It can be crude, buggy, incomplete, and poorly documented. What it must not fail to do is convince potential co-developers that it can be evolved into something really neat in the foreseeable future.”*

However, producing a plausible promise turned out to be more challenging than we expected. To date, a full six years after the VoiceCode project was started, only two other people have contributed code to it: David Fox (then an astronomy student at Harvard who now works at Nuance) and Stuart Norton (then a student at UC Santa Cruz who now works at Borland). Our biggest mistake was that we tried to get people excited about VoiceCode’s potential by focusing first on its coolest feature: the ability to translate pseudo-code utterances like *“for each index up to ten do the following”* into native

programming code. We demonstrated this feature early on, but not in a way that could be used for real. You had to simulate the act of speaking by typing text in a DOS console window, and the result appeared in a console window instead of inside an actual editor. While this gained us much kudos from the community of programmers-by-voice, it was obviously not perceived as a believable promise, because no-one actually contributed code to the project.

In my experience, people only start believing the promise when they can use the system for something real, even if it is small. Yet, if you are to co-opt people into investing countless hours in developing the system, it better be cool too. A good balance seems to be to *"Think Big and Cool, but start Small and Useful"*. In other words, write down a clear, big and cool vision, but start by implementing a very small end-to-end piece of it that people can use right away. With VoiceCode, we did Think Big and Cool, but we failed to start Small and Useful.

It's all about people and collaboration, not software

It has often been noted that in software development, people and collaboration issues are more critical than technical ones. Open Source development is no exception, and it some ways, it has developed practices that make collaboration easier. Indeed the corner stone of Open Source, the GPL license, was designed specifically to foster collaboration by allowing developers to build upon each other's work. Also, one of the most attractive things about Open Source for developers is that it allows them to engage in direct conversations with the users (usually through email), without having to go through an intermediary like a marketing department. Developers quickly become addicted to such direct feedback on their work.

While the GPL and direct contact with users help with collaboration issues, Open Source development presents special challenges of its own in that respect. Because team members often live miles away from each other and sometimes in different time zones, it is harder for them to coordinate. Moreover, because the project usually has no funds, it is not economically possible for team members to travel and meet face to face. Finally, the fact that lines of authority in the project are usually inexistent means that people have to work harder to agree on things and work towards a common goal.

With VoiceCode, we did a number of things to facilitate collaboration. Firstly, even before we started the project there existed a mailing list called VoiceCoder, founded by Brad Litterel in 1999, where programmers-by-voice discussed particular issues they faced and exchanged best practices. This forum proved invaluable for connecting with our constituency of future users, and to define a vision for the project. However, we found email contact to be somewhat limiting. Therefore in March 2000, Eric Johansson and I organized a one day face-to-face workshop in Boston (a hotbed for speech recognition) where 21 people interested in programming-by-voice met to discuss and exchange ideas. This workshop was instrumental in shaping up the VoiceCode White Paper [7], a document that laid down the vision which the team pursued throughout the project.

On a more day to day basis, we used frequent emails and a wiki site to coordinate. Combined with a clear common vision, this turned out to be fairly efficient, but not as efficient as face-to-face contact. At one point, David Fox and I kept engaging in heated technical debates that took several days and sometimes close to a hundred emails to resolve. Sometimes the tone of the messages bordered on flaming, which kept both of us awake at night. Eventually, we decided that the spirit of our collaboration would be greatly improved by having a weekly one hour meeting on the phone. This turned out to be instrumental in allowing us to keep a good working relationship and to discuss difficult issues more efficiently and in a spirit of mutual respect.

In summary. Make sure you have a good way to communicate with your constituency of future users very early on, ideally even before you have written a single line of code. Arrange for key players to sporadically meet face to face. Keep the team working towards a same goal by defining a clear vision that everyone can buy into right from the outset. Use lots of asynchronous communication (email, wiki) as the basic mechanism for communication within the team, but make sure you meet regularly in a more synchronous medium to foster good working relations.

Take a leaf from the Extreme Programming book

I knew from day one that an Open Source project would not be amenable to top-down waterfallish management. After all, how much control can you, as a leader, exert on a team of volunteer developers? But at the same time I knew that I would have to put some sort of lightweight process in place to control (and hopefully leverage) the somewhat chaotic nature of an Open Source project. I just didn't know what.

Then in 2002, I attended a one week training workshop on Extreme Programming (XP) [3]. I could immediately see how the practices of XP were immediately applicable to an Open Source project. In particular, practices like close collaboration with customers/users, Small Iterations, Release Planning, Simplicity (*"StartWithTheSimplestThingThatCouldPossiblyWork"*) and No Functionality Added Early provided a clear and operational way of rapidly delivering a "believable promise". I could also see how Test Driven Development, Collective Code Ownership, Coding Standards and System Metaphor would help newcomers in understanding the code and making them feel comfortable with it (something which is important for recruiting new contributors).

I shared what I had learned with my co-developer David Fox, and we immediately started applying some of the practices in VoiceCode. This turned out to be very useful. In some cases (ex: Test Driven Development), we had already been using embryonic versions of the practice. But XP showed us a way to "turn the dial to max". Altogether, using XP practices made such a difference that I found myself wishing I had learned about them before I had started the project. In particular, I am convinced that it would have focused us on delivering something smaller and useful end-to-end early on (probably within a year). Unfortunately, by the time I learned about XP, the system had already grown big and ambitious, and we felt compelled to complete what we had started.

In summary, before you start an Open Source project, learn as much as you can about Extreme Programming and other Agile Development [1] methods. This may save you a lot of headaches.

Conclusion

Dumping code on the SourceForge site does not an OpenSource project make. Running a successful Open Source is hard work. It requires that you pay a lot attention to delivering end-to-end value early on, ensuring collaboration amongst developers and with stakeholders, and developing light handed processes for managing and leveraging the somewhat chaotic nature of Open Source development.

References

[1] Agile Alliance. http://www.agilealliance.org/

[2] Désilets, A., Fox, D. C., Norton, S. "VoiceCode: an innovative speech interface for programming-by-voice". CHI '06 extended abstracts on Human factors in computing. Montréal, Québec, Canada, p. 239 - 242 , 2006.
Also available on this page:
http://iit-iti.nrc-cnrc.gc.ca/publications/author-auteur/desilets_alain_e.html

[3] Extreme Programming Collective. *"The Rules and Practices of Extreme Programming."* http://www.extremeprogramming.org/rules.html

[4] Graham-Rowe, D. "Software lets programmers code hands-free."
http://www.newscientisttech.com/article/dn9066.html

[5] Raymond, E.. "The Cathedral and the Bazaar".
http://www.firstmonday.org/issues/issue3_3/raymond/.

[6] The VoiceCode web site. http://voicecode.iit.nrc.ca/

[7] VoiceCode White Paper.
http://voicecode.iit.nrc.ca/VoiceCode/public/wiki.cgi?obj=VoiceCode_white_paper

[8] Proceedings of the 1st VoiceCode Design Session. Boston, March 19th, 2000.
http://voicecode.iit.nrc.ca/VoiceCode/uploads/VCode1stMeeting/

[9] VoiceCoder mailing list. http://groups.yahoo.com/group/VoiceCoder/.