Guía de supervivencia para contribuir al OSS

Leo Soto M.
leo.soto@gmail.com

9 de agosto de 2010

Resumen

Involucrarse en un proyecto open source no es tarea fácil. Dejando afuera los aspectos técnicos de cada proyecto, una dificultad adicional es entender la dinámica social, los códigos y las expectativas de la comunidad de desarrollo OSS. Esta charla mostrará las lecciones obtenidas por el autor en su camino como desarrollador participando en diversos proyectos open source.

Palabras Claves: Código Abierto, Comunidad open source, Desarrollo de software.

Introducción

Las reglas de la comunidad open source no están escritas en ningún manual definitivo. Existen guías y fuentes mas importantes que otras, pero la forma en que un desarrollador, tester, diseñador, traductor o cualquier otro miembro activo de un proyecto pasa a formar parte del mismo no está escrita.

La razón es muy simple: No existe *la* forma de hacerlo. Mas allá de variar de proyecto en proyecto, consiste también en una combinación de costumbres sociales que han evolucionado durante los años y que también *cambia* con los años.

Motivación

Es en estas circunstancias cuando la experiencia toma un valor adicional. Porque a través de la experiencia de quienes han aprendido ha relacionarse con la comunidad open source es como otros pueden aprender, o al menos hacerse una idea de como involucrarse en ella de manera más efectiva.

¿Pero qué quiere decir "involucrarse de manera más efectiva"?

Para este trabajo, el foco está en dos puntos: (a) hacer que los primeros contactos no sean frustrantes y (b) que los resultados sean relativamente inmediatos (i.e, conseguir que las contribuciones sean aceptadas por los proyectos en los que se desea colaborar)

El autor ha de reconocer humildemente que su introducción en el mundo del OSS no fue particularmente efectiva, fallando ambos criterios. Sin embargo, y mediante la aplicación sistemática (y no tan sistemática) de la prueba y error, sumada quizás a algo de madurez personal y muy probablemente acompañada de uno o dos golpes de suerte, el autor es hoy un miembro activo (y efectivo) en varios proyectos open source.

Lamentablemente, llegar a este punto tomó bastante tiempo y, en retrospectiva, el autor considera que de haber recibido ciertos *hints*, su introducción al mundo OSS hubiera sido mucho más efectiva.

Cansado de esperar la invención (¡y masificación!) de una máquina del tiempo que le permita corregir tamaña injusticia, por ahora el autor se conforma con difundir dichos *hints* en esta charla[1]. El objetivo es que *otros* puedan involucrarse en el open source de manera efectiva. Y tal vez incluso salvar a mas de alguien que en lugar de rendirse tras la primera frustración, consiga entender que detrás del aparentemente frío y huraño mundo del desarrollo open source existe suele existir una comunidad de personas tan amable como cualquier otra.

Desarrollo del Tema

En su calidad de charla más motivacional que técnica, el trabajo se centra en historias, anécdotas y lecciones consideradas como claves por el autor. La idea es evitar quedarse en lo abstracto y mostrar en concreto como los *hints* dados han aplicado en la práctica a proyectos reales. A continuación se describen los puntos principales a tratar, los cuales son entremezclados con historias y reflexiones que por razones de espacio no se pueden incluir en este documento.

Preguntas inteligentes

Uno de los primeros errores que uno puede cometer al interactuar con los desarrolladores de un proyecto OSS, es hacer las preguntas equivocadas. No hay nada de malo en hacer preguntas, siempre y cuando sean mas o menos inteligentes.

Afortunadamente, sí existe un excelente escrito al respecto, con el original título de "Cómo hacer preguntas de manera inteligente" [1]. Del contenido de dicho artículo, se hará énfasis en (a) intentar encontrar una respuesta *antes de preguntar*, (b) ser preciso e informativo *al preguntar*, (c) interpretar lo que se lee *cuando se recibe una respuesta*, y (d) Cómo no reaccionar.

Con esto se pretende cubrir rápidamente el "ciclo de vida" de una pregunta, sin repetir el contenido completo del artículo original pero sí dejando una idea de su contenido para que los interesados puedan luego leer el texto completo.

[1] Que por cierto fue expuesta con éxito en la FLISOL 2010

Escogiendo una primera contribución

Una excelente estrategia para empezar es contribuir algo (a) simple y (b) que tenga utilidad para uno mismo. Lo primero aumenta las probabilidades de poder implementar la contribución en cuestión y que la implementación sea correcta.Lo segundo le agrega una motivación práctica que asegura que la contribución está resolviendo un problema concreto y real. Además de que, en el caso que la contribución sea rechazada, siempre seguirá siendo útil para uno.

Inglés

Mejorar el inglés es una necesidad. De lo contrario uno se aisla en los pocos proyectos donde uno se puede dar el lujo de hablar solamente en español. Sin embargo, no es necesario tener un inglés británico perfecto. En realidad ni siquiera es tan necesario hablarlo. Basta con entender el inglés escrito y ser capaz de escribir decentemente.

Seguir el bug tracker

Una forma de ir aprendiendo cada vez más de cómo funciona un software es seguir el bug tracker y, aunque no se pueda aportar con código, reportar los errores que se encuentren. Una vez que alguien los arregle se podrá mirar cómo fue arreglado el problema, y se aprenderá del código. Y luego, cuando sea posible adjuntar parches a los reportes, los desarrolladores del proyecto muy probablemente tendrán una buena disposición con uno.

Identificar cuando se ha tenido éxito

Es muy simple saber que las contribuciones han caído en tierra fértil. Basta observar si, de parte de los desarrolladores del proyecto, se observan dos cosas: entusiasmo y colaboración. Si es así, aún cuando la contribución no haya sido aceptada a la primera, lo será más temprano que tarde.

Conclusiones

El cierre de la charla está dedicado a recapitular las lecciones obtenidas a lo largo de las historias y anécdotas. Lecciones que incluyen algunos de los puntos principales tratados, como hacer cosas que le sirvan a uno mismo y el perfeccionar el inglés. Y también lecciones de vida que van mas allá del open source como cultivar la paciencia, aprender a seguir antes que liderar y siempre ensayar, errar y aprender. Y lo más importante: Tener “cuero de chancho”.

Referencias

[1] Eric S Raymond, *Cómo hacer preguntas inteligentes*, `http://www.sindominio.net/ayuda/preguntas-inteligentes.html`

Área en la que se centra el trabajo: Desarrollo de software

Nivel: Básico

Enfoque de la presentación:

- Todos los puntos tratados en el desarrollo del tema.
- La actitud de "cuero-de-chancho" que es transversal a dichos puntos.
- Reflexiones intermedias, incluyendo:
  - Las consecuencias de que el open source permita que el código de uno sea usado de formas que uno nunca imaginó.
  - La relación entre crear un proyecto open source y crear un movimiento.

A. Acerca del Autor

A.1. Reseña

Leo Soto es Ingeniero Informático de la USACH, socio de Continuum y desarrollador para Hashrocket Chile. Como parte de su experiencia en el mundo OSS, es parte del del equipo de desarrollo de Jython desde el 2008, tras haber participado en el Google Summer of Code de ese mismo año. También es colaborador en Mongoid, un ODM para el lenguaje Ruby y MongoDB, una base de datos no relacional.
Ha dictado charlas fuera del país en la DjangoCon 2008 realizada en el GooglePlex en Mountain View, y en la PyCon 2009 en Chicago. En el medio nacional ha participado como expositor en las FLISOL 2009 y 2010, Encuentro Linux 2008 y 2009 y en las Jornadas Regionales del Software Libre 2009.

A.2. Datos de contacto

- **E-mail:** leo.soto@gmail.com
- **Teléfono: 9 8733 9008**
- **URL:** `http://blog.leosoto.com`