Dirección de Proyectos Fin de Carrera orientados a mejoras de grandes aplicaciones de código abierto

Juan A. Pereira Varela
Dpto. Lenguajes y Sistemas Informáticos
Universidad del País Vasco
Pº Manuel de Lardizabal, 1. 20018
juanan.pereira@ehu.es

Resumen

Durante el curso 06/07 dos alumnos de la Facultad de Informática de San Sebastián participaron con sus proyectos fin de carrera (PFC) en en el Google Summer of Code 2007, un concurso de programación de nivel internacional.
Ambos estudiantes finalizaron con éxito sus proyectos, llevándose cada uno 4.500 dólares de premio y el reconocimiento tanto de Google como de las organizaciones de software libre para las que implementaron el PFC (GNOME[11], Moodle [10]). Además de los premios, los alumnos tomaron parte en el desarrollo de proyectos software open source de miles de líneas de código y decenas de programadores dispersos por el mundo. Aprendieron a usar técnicas y herramientas de ingeniería del software en un entorno real, en aplicaciones con una base de miles de usuarios. Ésta experiencia no sería posible mediante un PFC de ninguna otra manera. Presentamos en este trabajo las posibilidades que tiene para el alumnado de PFC la posibilidad de participación de forma activa en la mejora de grandes proyectos de software libre, desgranando los beneficios que esto aporta, tanto a nivel de conocimientos reales de ingeniería del software como de motivación personal y económica, todo ello, permitiéndole obtener una excelente calificación académica.

1. PFCs y Software Libre

En la actualidad, la realización de Proyectos Fin de Carrera (en adelante PFCs) es obligatoria para obtener el título de ingeniero informático. Se busca que el alumno adquiera destrezas sobre ingeniería software y gestión de proyectos no triviales en el mundo real. Por otro lado, a día de hoy, el mundo del software libre ofrece la posibilidad de que personas con conocimientos informáticos avanzados (conocimientos como los que un ingeniero informático debería de tener) colaboren en los desarrollos de nuevas funcionalidades o nuevas aplicaciones software, requeridas por miles de personas de todo el mundo. Podría decirse que ambas necesidades forman parte de dos organismos en relación simbiótica, es posible que cada uno de ellos se aproveche del otro y como resultado, ambos obtengan enormes beneficios. El trabajo que se presenta pretende explorar esta relación de simbiosis, mediante el análisis de varios casos reales llevados a cabo durante el curso 2006/2007 en la facultad de Informática de San Sebastián.

En 2007, se planteó a dos alumnos de esa facultad la realización de dos PFC para dos grandes organizaciones del mundo opensource: GNOME y Moodle, respectivamente. En el primero, se planteó un proyecto de adición de una nueva funcionalidad para la creación

de anotaciones en ficheros PDF dentro del visor Evince. En el segundo, se planteó crear un nuevo plugin para la plataforma de eLearning, que permitiera a los profesores de asignaturas como "Programación I" recibir el código fuente de problemas planteados a los alumnos y que éstos pequeños programas fueran compilados y probados automáticamente, sin intervención del profesor. Ambos proyectos partieron con un triple objetivo inicial, de mayor a menor importancia, 1) que sirvieran como proyectos software de gran envergadura donde poner a prueba los conocimientos adquiridos durante la carrera (y otros nuevos, no vistos hasta el momento), 2) que los resultados fueran de utilidad para todos los usuarios de software libre a nivel mundial y 3) que el proyecto tuviera la complejidad suficiente como para presentarlo al Google Summer Of Code 2007 [4] (en adelante SoC07). En este concurso participaron más de 6.000 propuestas, siendo unas 900 las seleccionadas y de éstas, no todas terminaron satisfactoriamente, es decir, menos de 900 propuestas recibieron los 4.500 dólares de premio por parte de Google.

La documentación, el código fuente y cualquier otro artefacto generado bajo los PFC que en 2007 implementaron nuevas funcionalidades en aplicaciones libres, debían de licenciarse bajo una licencia aprobada por la Free Software Foundation [6], una licencia libre, de tal forma que los desarrollos pudieran ser utilizados por la comunidad opensource sin problemas de licencia.

Los dos alumnos mencionados consiguieron ser seleccionados para el SoC07, concluyeron con éxito sus asignaciones y aprobaron satisfactoriamente (en uno de los casos con Matrícula de Honor) sus PFC.

2. Proyectos internacionales de código abierto y PFCs: una relación de simbiosis

Durante el desarrollo de los proyectos opensource mencionados, los alumnos adquirieron y perfeccionaron multitud de habilidades, técnicas y capacidades que, a la sazón, son precisamente aquellas que se buscan en la definición de un PFC.

Desgranaremos a continuación algunas de las más importantes, haciendo especial hincapié además, en los efectos beneficiosos que aporta el haber optado por un proyecto opensource frente a otras alternativas clásicas.

2.1. Trabajo en problemas reales y de gran impacto

Los alumnos participaron en proyectos opensource no triviales, como puede verse en la tabla 1. Nunca antes habían trabajado en proyectos software de este calibre y uno de los puntos que agradecieron al finalizar el proyecto era, según sus propias palabras, el haberles facilitado esta oportunidad. Es de reseñar así mismo, la importancia que tienen las grandes aplicaciones del mundo del software libre a nivel de exposición al mundo real (cientos de miles o incluso millones de usuarios en el caso de aplicaciones libres como Apache, Firefox u OpenOffice). Cuando los alumnos hicieron públicas sus intenciones de desarrollo, recibieron multitud de peticiones y sugerencias para desarrollar las funcionalidades propuestas, así como casos de uso [3], ideas generales y ánimo Estas peticiones fueron consideradas como peticiones del cliente en un proyecto de software privativo tradicional, tomando al cliente como una gran empresa, de la que se reciben peticiones que hay

que saber filtrar (es decir, la vida real del ingeniero software). Aún a día de hoy siguen llegándoles este tipo de sugerencias y comentarios (Febrero 2008). Este tipo de feedback motivó que ambos PFC construyeran un blog desde el principio del desarrollo que, a día de hoy, aún mantienen actualizado [1].

Aplicación: EVINCE	**Valor**
Líneas de código	57.410
Número de desarrolladores	172
Esfuerzo estimado de desarrollo	14 pers/año
Aplicación: MOODLE	**Valor**
Líneas de código	536.623
Número de desarrolladores	126
Esfuerzo estimado de desarrollo	144 pers/año

Tabla 1. Datos relativos a la complejidad de aplicaciones Evince y Moodle. Obtenidos de http://www.ohloh.net/

La interacción con los usuarios genera una especial motivación en el alumno; hay muchas personas pendientes del trabajo a realizar, no sólo es un PFC al uso que una vez terminado, en ocasiones, acaba en la biblioteca de la facultad, sino que los resultados se usarán, y mucho, a nivel mundial. Por otro lado, tras lo dicho, es evidente que el idioma utilizado para comunicar avances, interaccionar con los grupos opensource y los usuarios de la aplicación ha sido y es el inglés. Lógicamente, este factor influye en la consideración o evaluación del tribunal del PFC, dado que no abundan los proyectos que usen este idioma, en exclusiva, a lo largo de todo el desarrollo.

Realizar PFCs basados en la adición de mejoras o nuevas funcionalidades a grandes proyectos opensource no sólo aporta beneficios para el alumno. El carácter muchas veces voluntario de la implicación del desarrollador en proyectos opensource suele acarrear que con el tiempo, otras tareas más urgentes, prioritarias o de mayor interés hagan que su implicación decaiga. Los propios proyectos opensource intentan por tanto atraer savia nueva y fomentan la implicación de nuevos desarrolladores constantemente. Un alumno de PFC es una opción ideal, dado que no sólo se implicará por gusto, sino también por obligación dado que ha de completar satisfactoriamente el PFC para poder graduarse. Ésto aporta al proyecto opensource un mayor grado de confianza en la nueva incorporación y cierta garantía de que durante varios meses, el nuevo desarrollador se implicará realmente en el proyecto, de forma constante.

3. Uso de herramientas de Ingeniería del Software

Trabajar en aplicaciones software donde más de 100 desarrolladores colaboran en la misma base de código, tal y como se ve en la tabla 1, es una tarea compleja que requiere de planificación, técnicas y herramientas adecuadas. Muchas de ésas técnicas y herramientas, tras ser vistas en teoría en asignaturas de la carrera, el alumno de PFCs opensource las podrá poner en práctica. Algunos ejemplos destacados podrían ser los que se describen en los siguiente puntos.

3.1. Herramientas de Planificación

La planificación de los proyectos opensource sigue en la actualidad procesos bien definidos. El alumno que participe en ellos debe de indicar inicialmente, con un documento descriptivo las diferentes fases en las que piensa desarrollar su trabajo, indicando grosso-modo, algunos detalles de lo que va a realizar - a nivel de diseño y código - en cada fase. A ser posible, ese documento descriptivo debería de ir acompañado de un

diagrama de Gantt, marcando claramente los hitos y enlazando dichos hitos con las fechas de publicación de versiones *alpha* y *beta* de la aplicación para la que se desarrolla el PFC. Los hitos (*milestones)* de una gran aplicación de código abierto (como Evince, o Moodle) han de ir sincronizados con otras aplicaciones en las que se apoyen. Por ejemplo, Evince debe de publicar nuevas versiones de forma sincronizada con la biblioteca *poppler*, en la que se apoya para obtener la información de ficheros PDF. Otras veces deben de ir sincronizados con procesos de publicación ya establecidos en el proyecto padre (en el caso de Evince, las versiones oficiales con grandes cambios se publican aprovechando la salida de nuevas versiones del escritorio GNOME - un entorno de escritorio Linux que abarca multitud de aplicaciones oficiales, entre ellas Evince)

3.2. Herramientas de gestión de errores (bugs)

En la facultad no se trabaja con herramientas de gestión de errores (bugs), tales como Bugzilla [2] o MantisBT [9], cuando en proyectos opensource (y también en proyectos de soft propietario) resultan imprescindibles para no solapar trabajos entre los desarrolladores, informar de avances, informar de problemas y por cuestiones de trazabilidad (saber quién realizó qué parche, o qué funcionalidad). El alumno de PFC opensource experimentará con estas herramientas a lo largo de todo su desarrollo.

Herramientas como Bugzilla o MantisBT no sólo se usan para informar o gestionar errores sino también para gestionar el desarrollo de nuevas funcionalidades. Se permiten añadir descripciones de nuevas funcionalidades como si de bugs o errores se trataran, especificando quién, cómo y cuándo debe de "arreglarse" (entiéndase para el caso de nuevas funcionalidades como "cuándo debe de programarse")

El proceso de dar de alta una nueva entrada en un sistema de gestión de errores puede verse como un proceso complementario a la planificación que se ha descrito anteriormente, dado que en esas entradas el alumno debe de especificar la versión de la aplicación en la que estima que su funcionalidad verá la luz.

3.3. Sistemas de control de versiones y calidad

Otra herramienta usada en grandes proyectos de ingeniería software son los sistemas de control de versiones. Dos de los más conocidos en la comunidad opensource son CVS [5], Subversion [12] o git [7]. Es en estos sistemas donde el alumno ha de trabajar diariamente, descargando los parches realizados por el resto de programadores y subiendo los suyos propios. En la comunidad del software libre, para asegurar la calidad del código, se establece que los sistemas de control de versiones no puedan recibir parches de desarrolladores no oficiales. Esto implica que, los alumnos de PFC no podrán "subir" directamente sus aportaciones de código, sino que han de ser aprobadas previamente por un grupo de desarrolladores de más alto rango, con permisos de escritura en el sistema. Este proceso se lleva normalmente a cabo a través de listas de correo: el alumno envía sus aportaciones junto a comentarios donde explica el porqué de cada decisión de diseño e implementación, y los desarrolladores oficiales aprueban o no dichas aportaciones o parches. En caso de no ser aprobadas, se explican las razones y se lanzan propuestas de mejora para que el alumno pueda corregirlas.

3.4. Documentación

Los trabajos a realizar por los alumnos han de disponer de una buena documentación en todas las fases del desarrollo, tanto porque así lo requieren las normas de desarrollo de un PFC como porque la comunidad opensource así lo pide. Esta documentación puede venir en forma de casos de uso en la web de la aplicación, en la propuesta de diseño inicial enviada para participar en el concurso SoC07, en el propio código fuente de la aplicación, en el sistema de control de errores o en el propio blog que los alumnos desarrollaron como complemento al PFC, donde explicaban sus ideas, diseños y planes.

4. Trabajo en grupo

Tal vez uno de los mayores desafíos. El desarrollo de funcionalidades sobre grandes aplicaciones opensource no sólo exige tener sólidos conocimientos de programación, sino también cierta dosis de ingeniería social. Es necesario saber tratar con otros usuarios y desarrolladores, en inglés, a nivel mundial. Es necesario saber cómo negociar, cómo convencer de las bondades de la idea del PFC, o cómo pedir ayuda cuando el alumno se atasca. Es necesario saber cómo comentar, sin herir susceptibilidades, los parches que el alumno u otra persona aporte (en ocasiones, rechazando un parche realizado por voluntarios por no cumplir con la calidad esperada). Se trabaja también la negociación de fechas de publicación de los aportes realizados, la corrección de forma comunitaria de problemas de integración, etc. Los grupos de desarrolladores en la comunidad del software libre suelen tener fuertes normas en lo que respecta a la calidad del código fuente; parches de código funcionales desarrollados por los alumnos de PFC, que por alguna razón no seguían las normas del grupo o las ideas de diseño que el grupo había establecido supuso algunos retrasos. Rehacer parte del trabajo o bien acordar alguna solución alternativa óptima para todos fueron algunos de los acuerdos que se tomaron. Este proceso, en general, se lleva a cabo bien por correo electrónico o bien por medio de reuniones virtuales vía IRC (chat).

Otra de las tareas "sociales" que los alumnos han tenido que practicar es la de buscar el apoyo de personas para aprobar los parches que iban proponiendo a los desarrolladores principales. Este hecho se debe a que los desarrolladores principales suelen ser pocos y en la mayoría de las ocasiones, el desarrollo de las aplicaciones libres no es su principal función, lo que provoca que otras tareas de mayor prioridad hagan que las relacionadas con el PFC no sean tratadas . Encontrar a desarrolladores o usuarios colaboradores de los desarrolladores principales y motivarlos para que pidan que se aprueben los parches ofrecidos por el alumno ha sido otra de las denominadas tareas de "ingeniería social".

5. Otros efectos beneficiosos

Las estadísticas de la dirección de proyectos PFC muestran que es frecuente que los alumnos abandonen el proyecto y no lo defiendan el mismo año lectivo en el que lo comienzan. Algunas de las razones que aducen los alumnos son la pérdida de motivación y la mala planificación (en ocasiones reconociendo su falta de autodisciplina).
¿Qué pueden aportar los PFC opensource a esto?

5.1. Motivación y disciplina

Los PFC relacionados con proyectos opensource, en el que los alumnos ven que los propios usuarios son los que alientan el desarrollo (con comentarios, críticas constructivas e incluso en ocasiones con trozos de código), provocan que el alumno reciba constante feedback o retroalimentación, lo que sirve como motivación diaria. La autodisciplina se hace más llevadera cuando el alumno está motivado.

Por otro lado, dado que los PFC que se discuten se presentan a concursos de programación como el Google Summer of Code o el Concurso Universitario de Software Libre [8], el saber que el esfuerzo de desarrollo puede ser premiado (con un certificado y una bolsa económica de ayuda de 4.500 dólares el SoC y con 2.000 euros el Concurso Universitario) es un acicate motivador.

5.2. Reconocimiento a nivel mundial (prestigio y curriculum)

La recompensa económica (sustancial para los alumnos de PFC) no es el único aliciente. Durante el curso 06/07 los alumnos participantes en el SoC fueron entrevistados por distintos medios de comunicación (periódicos *DiarioVasco* y *20Minutos*, revista *Campusa* de la Universidad del País Vasco) y se publicaron numerosas referencias en las webs de la propia facultad, de su Universidad y de las principales web de tecnología de la comunidad . Esta cobertura mediática, reconociendo el trabajo de los alumnos seleccionados para el concurso de programación de Google provocó en los mismos un efecto motivador aún mayor del que ya tenían al saberse seleccionados. Finalmente, el certificado de Google reconociendo el trabajo realizado ha enriquecido notablemente el curriculum de los alumnos participantes.

5.3. Aprender de los errores

Los alumnos saben que no siempre es posible ser seleccionado en un concurso internacional, y mucho menos conseguir llevar hasta el final planificado un gran proyecto software. Los alumnos del curso 06/07 lo consiguieron, pero en el curso 07/08 un alumno seleccionado para participar en el II Concurso Universitario de Software Libre ha visto que su planificación no era correcta y ha postpuesto su participación. Otro que comenzó motivado para participar en el SoC08 (si se volviera a organizar) ha perdido la motivación al ver que los conocimientos requeridos distan mucho de los que posee actualmente. Ambos alumnos han aprendido de los errores, se han enfrentado a grandes proyectos de ingeniería software y han sufrido las dificultades que éstos entrañan, sin que ésto haya tenido repercusiones económicas ni de otra índole en ninguna empresa (que sí hubieran tenido en caso de tratarse de PFC convencionales para empresas de software propietario u otras).

6. Consideraciones sobre la elección de proyecto

Los PFC de la Facultad de Informática de San Sebastián tienen como condición que la carga horaria mínima planificada sea de aproximadamente 150 horas en la Ingeniería Técnica y de 300 horas en la Ingeniería Superior.. Los dos proyectos presentados al Google Summer of Code 2007 desde dicha facultad tuvieron una planificación inicial acordada con los alumnos (de Ing. Técnica en ambos casos) de 300 horas. Esa carga horaria incluía la formación inicial en algunas de las tecnologías usadas en los proyectos opensource a mejorar. Es

evidente que este tipo de proyectos requieren por tanto un compromiso previo de trabajo duro por parte del alumno, pues en el caso de la Ing. Técnica, se les solicita invertir el doble de horas que otro PFC que se ajuste a una política de mínimos. Respecto a la planificación inicial hubo una desviación final de alrededor de un 21% respecto a esa cifra (426 horas). Las causas de ese desvío se debieron a la necesidad de completar la formación en APIs de desarrollo desconocidos y en dificultades para familiarizarse con el código fuente de las aplicaciones opensource que se pretendía mejorar. Los alumnos no disponían de experiencia previa en herramientas de gestión software (sistemas de control de bugs, sistemas de control de versiones, sistemas de pruebas unitarias, ...) aunque sí tenían conocimientos sólidos de desarrollo en C y PHP (lenguajes fundamentales para los proyectos seleccionados en nuestro caso). La dificultad del diseño y programación de las nuevas funcionalidades realizadas fue contrastada previamente con responsables del proyecto opensource. Por ejemplo, en el caso del soporte de anotaciones a documentos PDF en Evince, nos pusimos en contacto con uno de los desarrolladores principales, enviándole el análisis de lo que se pretendía programar, solicitando su validación. Este desarrollador nos validó la propuesta, nos confirmó que las horas estimadas se encontraban dentro de lo razonable, nos guió en la primera fase del diseño y nos ofreció su ayuda para cualquier problema que pudiera surgir.

A modo de resumen se pueden indicar, por tanto, dos puntos fundamentales a la hora de realizar la asignación entre PFC y la mejora de un proyecto opensource. Por un lado es necesario filtrar a los alumnos candidatos en función de sus conocimientos previos de programación y herramientas de ingeniería software. Por otro lado, es vital contar con la ayuda de un desarrollador veterano dentro del proyecto opensource, con el fin de validar que la dificultad de la tarea a emprender casa con lo que puede esperarse de un PFC y que el análisis y diseño preliminar es viable y coherente.

7. Conclusiones y trabajo futuro

Se han expuesto las ventajas de promover y promocionar el desarrollo de proyectos de software libre como Proyectos Fin de Carrera. Esta posibilidad no ha sido viable hasta hace poco tiempo, cuando el software libre ha alcanzado una etapa de madurez y calidad global, por lo que esta vía de desarrollo aún no ha sido suficientemente explorada ni explotada en los PFC. El profesorado debería de analizar las posibilidades que ofrece la colaboración entre la comunidad del software libre y la comunidad universitaria, con el fin de mejorar u optimizar los beneficios académicos y profesionales que el alumno puede obtener gracias a su trabajo en el proyecto fin de carrera.

Varias universidades nacionales (Universidade da Coruña, Universidade de Santiago, Escuelas de la Universidad de Castilla La Mancha (EPSA, ESI y la EUPC,...), han optado por no sólo colaborar ofreciendo alumnos sino patrocinando dichos concursos, siempre que los trabajos realizados sean licenciados bajo una licencia libre. Esta colaboración podría extenderse en un futuro entre más universidades, aprovechando, de forma comunitaria, no sólo las aplicaciones libres que se desarrollen, sino también los recursos asociados generados. Por ejemplo, compartir la documentación generada en el proyecto, tanto para ser

usada como material docente como base de históricos con el fin de comparar y cotejar. Otro punto de mejora significativo podría ser el de acordar entre varias universidades la mejora de una única aplicación anualmente, entre PFCs de todo el país, de tal forma que las mejoras recibidas contribuyeran a elevar la calidad y funcionalidad global de dicha aplicación de forma sustancial.

Referencias

[1] Blog del alumno participante en el Summer of Code 2007 desarrollador de Evince: annotations.diariolinux.com
[2] Bugzilla, sistema de gestión de errores: www.bugzilla.org
[3] Casos de uso de la funcionalidad “Anotaciones” en el visor de PDFs Evince annotations.diariolinux.com
[4] Concurso internacional de programación de aplicaciones opensource Google Summer of Code. code.google.com/soc
[5] CVS, sistema de control de versiones, www.nongnu.org/cvs
[6] Free Software Foundation, www.fsf.org
[7] git, sistema de control de versiones, git.or.cz
[8] II. Concurso de Software Libre Universitario, concurso-softwarelibre.us.es
[9] MantisBT, Sistema de gestión de errores: www.mantisbt.org
[10] Moodle, sistema de eLearning con licencia opensource, www.moodle.org
[11] Proyecto GNOME, www.gnome.org
[12] Subversion, sistema de control de versiones, subversion.tigris.org