Participación de los estudiantes en una comunidad de software libre como criterio de evaluación

Ricardo Medel

Departamento de Ingeniería en Sistemas de Información
Universidad Tecnológica Nacional – Facultad Regional Córdoba
Maestro M. Lopez esq. Cruz Roja Argentina, Ciudad de Córdoba, Argentina
rmedel@frc.utn.edu.ar

Resumen

El sofware libre es creado por comunidades de desarrolladores voluntarios, usuarios y empresas distribuidos globalmente. Debido a que puede ser utilizado, modificado y distribuido libremente por cualquier persona o entidad con cualquier propósito, el software de este tipo tiene gran difusión, tanto en ambientes educativos como comerciales e industriales. Nuestra universidad, sin embargo, tiene una larga tradición de no utilizarlo. Es por esto que desde el año 2015 se ofrece una asignatura electiva sobre software libre a estudiantes del último año de la Ingeniería en Sistemas de Información. La principal innovación es que, como parte de los criterios de evaluación del curso para su aprobación, los estudiantes deben participar activamente en una comunidad de software libre de su elección. En este trabajo reportamos los resultados, en general positivos, de esta exposición de los alumnos a la realidad del ambiente de desarrollo y utilización del software libre.

Palabras clave: *Software libre, Metodologías de desarrollo, Ingeniería de Sistemas de Información*

1. Introducción

El software libre y su similar, el software de código abierto o de fuentes abiertas, son tan antiguos como las computadoras. En los inicios de la era de las computadoras, el valor comercial de un sistema provenía solo de hardware, mientras que el software era compartido libremente por las usualmente pequeñas comunidades de desarrolladores y operadores creada alrededor de un hardware en particular (Dibona, Ockman y Stone, 1999). Recién en 1984 Richard Stallman definió formalmente el término "software libre" en El Manifiesto de GNU (Stallman, 2016). Más tarde, la creación de Open Source Initiative aportó la definición de software de código abierto (OSI, 2007). A pesar de algunas diferencias entre el software libre y el software de código abierto, ambos comparten la misma filosofía de construcción colectiva, libertad de uso y modificación colaborativa del código.

Técnicamente, el software libre no se diferencia del software propietario (aquel software cuya licencia de uso no es libre). Sin embargo, su gran diferencia es la estructura de desarrollo y mantenimiento que se genera alrededor de cada proyecto de software libre. Incluso cuando algunos de estos proyectos son guiados por empresas, la mayoría de los proyectos son creados y mantenidos por una comunidad de desarrolladores y usuarios siguiendo distintos modelos de organización (Raymond, 2009).

Este enfoque novedoso para la construcción de software, así como sus aspectos filosóficos, sociales, legales y económicos, requieren que los profesionales de software estén informados sobre sus características, ventajas y limitaciones. Este conocimiento les permitirá tomar decisiones informadas sobre el desarrollo y uso de software libre durante su práctica profesional.

Debido a su bajo o nulo costo, y a la posibilidad de modificar y distribuir libremente el código fuente, este tipo de software se utiliza comúnmente en educación, especialmente en universidades con programas de informática o ingeniería de software (Lakhan y Jhunjhunwala, 2008;

Wilson, 2013). Sin embargo, nuestra Universidad tiene una larga tradición en no utilizarlo. En los pocos cursos donde se lo utiliza, sus características no técnicas (filosofía, licenciamiento, comunidades de desarrollo, etc.) no se estudian ni analizan.
Es por esto que desde el año 2015 se ofrece una asignatura electiva sobre software libre a estudiantes del último año de la Ingeniería en Sistemas de Información (Cabral et al., 2014; Medel, 2016).
Además del objetivo de ampliar el cuerpo de conocimientos de los futuros ingenieros de sistemas de información, un aspecto crucial de esta propuesta es que, como parte de los criterios de evaluación del curso para su aprobación, todos los estudiantes deben elegir una comunidad de software libre ya existente y participar activamente en ella. En este trabajo presentamos los resultados de nuestras experiencias durante los cuatro años en que se ha ofrecido el curso. Cabe destacar que este curso no fue diseñado como un experimento sobre Educación, en cuyo caso requeriría un grupo de control y evaluaciones previas y posteriores de los estudiantes. Nuestro objetivo aquí es más bien modesto: comunicar las lecciones aprendidas durante la implementación de este curso en la carrera de Ingeniería de Sistemas de Información, con la esperanza de ser útiles para otras universidades o carreras que estén en situaciones similares.

2. Marco teórico

El "software libre" es aquel software que asegura al usuario cuatro libertades esenciales (FSF, 2018):

0. La libertad de ejecutar el programa como se desee, con cualquier propósito.
1. La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que el usuario quiera (el acceso al código fuente es una condición necesaria para ello).
2. La libertad de redistribuir copias para ayudar a otros.
3. La libertad de distribuir copias de sus versiones modificadas a terceros (el acceso al código fuente es una condición necesaria para ello).

Desde hace décadas, el software libre está presente en una gran cantidad de sistemas informáticos (Feller et al., 2005). En particular, fue y es clave en la infraestructura de Internet (Newman, 1999), y se puede encontrar incluso en los teléfonos inteligentes que usamos diariamente (Hoffman, 2014).
Como se dijo, debido a su bajo o nulo costo, a su cada vez mejor desempeño técnico y a la posibilidad de distribuirse libremente entre la comunidad educativa en general, el uso de software libre en educación se ha expandido en todos los niveles (García-Domínguez, Rodríguez Galván y Palomo-Duarte, 2008; Sánchez Vera, 2010). Esta expansión también se debe al incremento en la demanda de profesionales formados en software libre (Neira y Palomo-Duarte, 2008) y a la posibilidad de crear Recursos Educativos Abiertos (Meiszner, Glott y Sowe, 2008).
El participar en el desarrollo de software libre permite a los alumnos experimentar el proceso de desarrollo de software *in-the-large*, a diferencia de la experiencia usual *in-the-small*, la cual es realizar pequeños programas de práctica, con requerimientos, usuarios y demás restricciones ficticias, provistas por la cátedra. Es decir, permite a los estudiantes contribuir a la producción de software real, en un ámbito de desarrollo real (Buffardi, 2015; Wachenchauzer, 2013).
Desde 2010 las conferencias POSSE (Professors Open Source Software Experience) impulsan la participación de alumnos en proyectos de software libre humanitarios (HFOSS: Humanitarian Free/Open Source Software), cuyo principal objetivo es social (Dziallas, 2012; Ellis, Morelli e Hislop, 2008; Morelli et al., 2009).
Los resultados de estas experiencias evidencian mejoras en las habilidades técnicas y sociales de los estudiantes, aunque se requiere un esfuerzo extra de los profesores y, en particular, de los alumnos al involucrarse en una comunidad y realizar un aporte significativo (Pinto et al., 2017).
El principal motivador, desde el punto de vista de los alumnos es el poder aplicar habilidades de desarrollo de software a proyectos de la vida real, para clientes

reales, en un ambiente de producción, todo dentro de los límites de un semestre. Sin embargo, los desafíos no son menores: ¿se conseguirán los objetivos de aprendizaje? ¿los proyectos son adecuados para el nivel de los estudiantes? ¿utilizan lenguajes de programación que el estudiante conoce? (Hislop, 2009). En el caso argentino debemos sumar la cuestión del lenguaje de comunicación entre miembros de la comunidad, que usualmente es el inglés. ¿Podrán nuestros alumnos comunicarse en ese idioma?

3. Objetivos y Metodología

Como se estableció previamente, el objetivo de esta asignatura es completar la formación del egresado de la carrera a fin de que sea capaz de tomar decisiones informadas sobre el uso y desarrollo de software libre.

La asignatura se desarrolla durante el primer cuatrimestre, con una duración de 35 horas de clases teórico-prácticas (una clase de dos horas y media por semana).

Si bien hubo variaciones en el orden y profundidad a lo largo de los años, los temas que se estudian son los siguientes.

- Conceptos básicos del software libre
- Historia del software libre
- Licencias de software
- Gestión de comunidades de software libre
- Calidad de proyectos de software libre
- Modelos de negocios basados en software libre
- Sistemas operativos libres
- Reglas de diseño y programación aplicadas en comunidades de software libre
- La filosofía del software libre en otros ámbitos: educación, movimientos sociales y gobierno.

La asignatura ofrece aprobación directa con una nota igual o superior a 7 (siete), tomada como el promedio de los resultados de las evaluaciones aprobadas del alumno.

Las evaluaciones se basan en:

- Un examen parcial teórico-práctico y su correspondiente recuperatorio.
- Un informe escrito sobre la experiencia del estudiante en una comunidad de software libre.
- Una presentación oral sobre la experiencia del estudiante en dicha comunidad.

Mientras que el informe escrito es individual, la presentación oral puede compartirse entre alumnos que han participado de la misma comunidad.

La selección de la comunidad la realiza libremente cada alumno, aunque el equipo de cátedra provee un listado de comunidades (en las que se han realizado participaciones exitosas o se tiene cierta relación con la comunidad) que sirven como sugerencias.

En los primeros años se requería al alumno la realización de algunos ensayos cortos, a modo de ejericicios prácticos, dejando los dos últimos meses del cuatrimestre para la práctica en la comunidad. Desde el año 2017 se decidió eliminar dichos ejercicios prácticos y fortalecer la práctica comunitaria, comenzando antes el proceso de selección e inserción en la comunidad.

Por otra parte, desde ese año la asignatura se desarrolló aplicando la modalidad de “aula invertida”, lo cual no afecta a las prácticas en las comunidades de software libre.

4. Resultados

En esta sección resumimos los resultados de esta asignatura respecto de la cantidad de alumnos, las comunidades donde se realizaron las prácticas y la naturaleza de dichas prácticas.

A fin de cuantificar los resultados, mostramos en la Tabla 1 la cantidad de alumnos inscriptos, la cantidad de alumnos que comenzaron la asignatura y la cantidad de alumnos que obtuvieron la aprobación directa en cada año.

Tabla 1: Cantidad de alumnos.

Año	Inscriptos	Asistentes	Aprobados
2015	31	19	14
2016	29	13	12
2017	53	42	33
2018	45	37	27

Creemos que es necesario diferenciar entre las dos primeras columnas, ya que la primera incluye alumnos que sólo se inscribieron y nunca asistieron a ninguna clase de la asignatura. De esta manera, consideramos el índice de aprobación como la relación entre la cantidad de alumnos que efectivamente asistieron al menos a una clase y la cantidad de alumnos que aprobaron el curso.
Por su parte, la Figura 1 muestra los mismos datos en forma gráfica, lo que permite ver la variación en la cantidad de alumnos inscriptos, asistentes y aprobados a lo largo del tiempo.

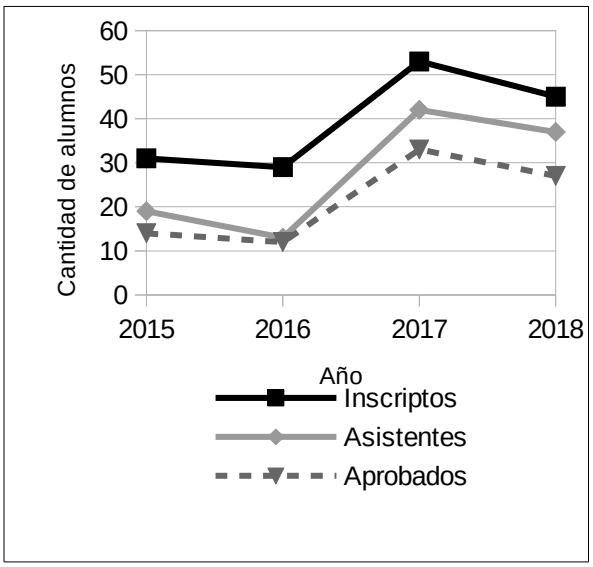

Figura 1: Cantidad de alumnos.

Respecto a la participación de los alumnos en las comunidades de software libre, la Tabla 2 indica la comunidad, la cantidad de alumnos que participaron a lo largo de todos los años de la asignatura y el tipo de actividades realizadas.
Se indica con ES las comunidades cuyo principal idioma de comunicación es el español, y con L las comunidades que son predominantemente locales, con la mayoría de sus integrantes habitando en nuestra ciudad. Consideramos relevante diferenciar la participación en comunidades globales respecto de las participaciones en comunidades locales, pues en ambos casos la problemática de la comunicación es diferente.

5. Discusión

Realizamos un análisis de los resultados de esta experiencia teniendo en cuenta la satisfacción de los alumnos, el índice de aprobación, la variedad de comunidades en las que se participó y el nivel de participación.
Al finalizar cada cuatrimestre se realizaron encuestas informales requiriendo la opinión de los alumnos respecto del curso y sus propuestas de mejoras. Las respuestas fueron mayormente positivas, mientras que las mejoras propuestas se centraron en extender el tiempo de la intervención en las comunidades (algo que se comenzó a hacer desde 2017) y en ofrecer el curso como asignatura electiva en tercer año. Esto tiene como motivo dos aspectos: por un lado, al aprobar tercer año los alumnos reciben el título de Analista en Sistemas y muchos de ellos no continúan la carrera de ingeniería; por otro lado, algunos alumnos indicaron que tener suficiente conocimiento sobre software libre durante el cursado de los últimos años de la ingeniería les hubiera permitido aprovechar mejor algunas de las enseñanzas impartidas.
El índice de aprobación, como lo muestra la Tabla 1, es muy elevado, rondando entre el 73% (2018) y 92% (2016).
La Tabla 2 muestra la variedad de comunidades en las que participaron los alumnos y la complejidad de intervención alcanzada. Si bien hubo muchas participaciones realizando traducciones y documentación, consideradas entre las actividades más básicas en un proyecto, hubo un importante número de participaciones que implicaron la corrección de errores o inclusión de nuevas funcionalidades, actividades que requieren de habilidades de programación y pruebas de calidad.
Sin embargo, más allá del nivel de participación, es importante recalcar que participar por primera vez de una comunidad de software libre requiere obtener un nivel de confianza en la comunidad que es difícil de lograr en apenas dos meses.

Tabla 2. Participación en comunidades de software libre

Comunidad	Alumnos	Actividades
pilas-engine (ES)	17	Refactorización de tests, desarrollo de juegos de ejemplo, reporte y corrección de errores, creación de fondos de pantalla, documentación, creación y actualización de tutoriales
PseInt (ES)	7	Creación de tutorial y FAQ, documentación
Node-open-pilot (ES, L)	6	Traducción al inglés y mejora de página web, agregado de funcionalidades, mejoras en el código
Reconocedor de billetes (ES, L)	6	Mejora de performance, reporte y corrección de errores, documentación de código
OpenSource.Guide	5	Traducción al castellano
PumaScript (ES, L)	5	Agregado de tests, documentación, nuevas funcionalidades
Hola	3	Traducción al castellano
OpenStreetMap	3	Agregado de información, traducción al castellano
Pilas Bloques (ES)	3	Desarrollo de capacitación
BabelZilla	2	Traducción al castellano, reporte de errores
Google Search API	2	Corrección de errores de instalador, reporte, revisión y corrección de errores
Ícaro (ES, L)	2	Desarrollo de placa ejemplo, documentación, traducción al inglés
Tails	2	Traducción al castellano
Airesis	1	Documentación
ASP .Net Core	1	Reporte de errores
CIAA (ES)	1	Mejoras en página web
Elastic Search	1	Detección de errores
Epoptes	1	Creación de tutorial
FullCalendar	1	Reporte de errores, nueva funcionalidad
HaxeFlixel	1	Desarrollo de juego de ejemplo
Koha	1	Traducción al castellano
LibreOffice	1	Reporte de errores
Match-it	1	Mejoras de performance
Mindfulness at the computer	1	Traducción al castellano, mejora de procesos, reporte y corrección de errores
Moodle	1	Traducción al castellano, reporte de errores
MySQL Workbench	1	Reporte de errores
Nelson	1	Traducción al castellano
Ninja-IDE (ES)	1	Mejora de procesos
OkHttp	1	Corrección de errores, mejora de procesos
Rayo.js	1	Traducción al castellano, mejoras en página web
Red-DiscordBot	1	Corrección de errores, traducción al castellano
REngine	1	Nueva funcionalidad y fork del proyecto
rst2html5 (ES)	1	Corrección de errores
Sakai	1	Detección de errores
Terasology	1	Agregado de música
VLC Media Player	1	Traducción al castellano

6. Conclusiones

En base a nuestro análisis consideramos que los objetivos fueron cumplidos: exponer a los alumnos a una práctica real de proyectos de software en ambientes de producción, al tiempo que se aprenden los principios técnicos, legales, filosóficos y sociales del software libre. Nuestros próximos pasos son realizar una evaluación más profunda de la participación de los alumnos y las características de tales comunidades a fin de seleccionar las mejores comunidades para participar y, en caso de nuevas comunidades, tratar de anticipar si la participación será enriquecedora. Por otra parte, continuaremos planteando a las autoridades la necesidad de ofrecer esta asignatura como electiva en tercer año o antes.

Referencias

Buffardi, K. (2015) *Localized open source collaboration in software engineering education*. IEEE Frontiers in Education Conference (FIE).

Cabral, J.B., Medel, R., Navarro, N., Reingart, M. (2014) *Propuesta de incorporación de la Ingeniería de Software Libre y de Código Abierto al currículo de Ingeniería en Sistemas de Información y carreras afines*. 1° Simposio Argentino de Tecnología y Sociedad, 43° JAIIO.

DiBona, C., Ockman, S., Stone, M. (1999) *Open Sources: Voices from the Open Source Revolution.* O'Reilly Media.

Dziallas, S., Ellis, H., Chua, M., Huss-Lederman, S., Wurst, K. (2012) *Teaching open source: involving students in free and open source software (FOSS) project communities (abstract only)*. 43rd ACM technical symposium on Computer Science Education, page 676.

Ellis, H., Morelli, R., Hislop, G. (2008) *Work in progress - Challenges to educating students within the Community of Open Source Software for Humanity*. Frontiers in Education Conference (FIE).

Feller, J., Fitzgerald, B., Hissam, S.A., Lakhani, K.R. (2005) *Perspectives on Free and Open Source Software*. The MIT Press.

FSF (2018) *¿Qué es el software libre?* Free Software Foundation.

García-Domínguez, A., Rodríguez Galván, J.R., Palomo-Duarte, M. (2008) *El software libre en el EEES.* Innovación Educativa para la Educación Superior: Hacia el Proceso de Convergencia. Madrid.

Gittlen, S. (2016) *6 Colleges Turning Out Open Source Talent*. Network World.

Hislop, G., Ellis, H., Tucker, A., Dexter, S. (2009) *Using open source software to engage students in computer science education*. 40th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2009. ACM SIGCSE Bulletin. 41. 134-135.

Hoffman, C. (2014) *Android is Based on Linux, But What Does That Mean?* How-to-Geek.

Lakhan, S.E., Jhunjhunwala, K., (2008) *Open Source Software in Education.* EDUCAUSE Quarterly. 31 (2).

Medel, R. (2016) *Experiencia del primer año de dictado de una asignatura electiva sobre Software Libre en la carrera de Ingeniería en Sistemas.* 3° Simposio Argentino de Tec. y Sociedad, 45° JAIIO.

Meiszner, A., Glott, R., Sowe, S.K. (2008) *Free / Libre Open Source Software (FLOSS) communities as an example of successful open participatory learning ecosystems*. European Journal for the Informatics Profession, UPGRADE. IX. 3, 62-68.

Morelli, R., Tucker, A., Danner, N., de Lanerolle, T., Ellis, H., Izmirli, O., Krizanc, D., Parker, G. (2009) *Revitalizing Computing Education Through Free and Open Source Software for Humanity*. Communications of the ACM. 52. 67-75.

Neira, P., Palomo-Duarte, M. (2009) *Innovación educativa con software libre.* VI Jornadas Int. de Innovación Universitaria.

Newman, N. (1999) *The Origins and Future of Open Source Software: A Net-Action White Paper*.

OSI (2007) *Open Source Definition*. Open Source Initiative.

Pinto, G., Figueira Filho, F., Steinmacher, I., Gerosa, M.A. (2017) *Training Software Engineers Using Open-Source Software: The Professors' Perspective*. 30th IEEE Conference on Software Engineering Education and Training (CSEE&T).

Raymond, E. (2009) *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly.

Sánchez Vera, M. (2010) *Los desafíos de la cultura participativa: Software libre y universidad*. Edutec: Revista electrónica de Tecnología Educativa. 33. 9.

Stallman, R. (2016) The GNU Manifesto. Free Software Foundation.

Wachenchauzer, R. (2013) Trabajos de Carreras de Informática en Comunidades de Código Abierto. Jornadas Argentinas de Software Libre, pp. 130-140.

Wilson, S. (2013) *Open source in higher education: how far have we come?* The Guardian.