Propuesta de lineamientos para la gestión de la cooperación en proyectos de desarrollo de software libre.

Ing. Alexander Betancourth Moncada, Dr. Jorge Andrick Parra Valencia Ph.D., M.Sc., SE,

Grupo de investigación Pensamiento Sistémico, Universidad Autónoma de Bucaramanga UNAB

webmaster@pereiravirtual.com, japarra@unab.edu.co

Abstract– Esta investigación es un estudio de los diferentes mecanismos de cooperación utilizados en algunos proyectos representativos de software libre (Kernel GNU/Linux, FreeBSD,Gentoo/KDE, Joomla-CMS, PhapMyAdmin y Ruby on Rails) con la pretensión de aportar lineamientos para aplicar en proyectos de desarrollo de software libre, de forma que la gestión de la cooperación en estos proyectos sea mejorada notablemente, mitigando problemas visibles como la deserción, la falta de incentivos y la división de la comunidad.

I. Introducción

En esta investigación se efectúa un análisis de producción de proyectos de software libre mediante la observación de comportamientos en la vida de los casos representativos estudiados, efectuando comparaciones en diferentes circunstancias del historial de producción tales como amplias pausas en los ritmos de producción, picos y valles prolongados de actividad reconocibles y valorables en la totalidad del proyecto. Relacionando estos comportamientos con datos cualitativos y cuantitativos que hacen parte de la caracterización de cada proyecto de desarrollo de software libre como: cantidad de desarrolladores, versiones publicadas y niveles individuales de actividad en los miembros en el ranking de más colaboraciones al interior del proyecto.

II. Los dilemas sociales y el software libre

Los dilemas sociales son situaciones en las que el individuo ve las elecciones que toma como la mejor opción entre el interés propio y el bienestar grupal. Son esas situaciones en las que individualmente las ideas y actos propenden por lograr el bien personal, produciéndose varias opciones para el individuo como es la de colaborar o la de no colaborar, también existe la opción de colaborar parcialmente; es así como la situación colectiva se vuelve dependiente de la motivación y acción individual, por lo general el individuo es poco consiente de la dependencia de la situación colectiva con respecto a las decisiones e intereses personales.

Existen varios ejemplos de la teoría de juegos que esquematizan las diferentes situaciones de intereses personales de los individuos de un grupo, uno de ellos es el dilema del prisionero, en este se plantea la necesidad de colaborar grupalmente con una decisión unificada o el castigo por elegir obtener mejor retribución que su compañero.

En la teoría de juegos se estudia detenidamente el equilibrio de NASH, que describe una situación en la cual los individuos poseen una estrategia, conociendo con anterioridad la estrategia de los otros individuos teniendo de esta forma la mejor manera de proceder en la situación, siendo poco o nada lo que ganaran si cambian su estrategia.

Podemos citar entonces comúnmente en los dilemas sociales algunas variables comunes para los problemas típicos, aquellas son:

- Elección individual
- Elección grupal
- Estrategia individual
- Estrategia de cooperación

Cada una de estas variables describe una parte del comportamiento colectivo en los dilemas de sociales de cooperación.

III. Los dilemas sociales de gran y pequeña escala

Los dilemas sociales independientemente de su escala poseen la similitud de motivación para el individuo, pero es de notar que los dilemas de pequeña escala difieren sustancialmente en las situaciones de dilemas de gran escala.

Existen diferencias muy claras en la dinámica de cooperación en dilemas de gran escala con respecto a los de pequeña escala podemos decir que los de pequeña escala poseen la facilidad para el individuo de racionalizar fácilmente su comportamiento, motivación y resultados gracias a la observación de la colectividad. También hace que cada individuo pueda comprometerse con los intereses de la colectividad analizando a su vez la interdependencia que poseen mutuamente los individuos y el grupo de pequeña escala; Pero no pasa lo mismo con los dilemas de gran escala en los cuales si bien la motivación es la misma para los individuos y el colectivo, no son tan directamente observables las consecuencias como producto de un aporte individual, podemos tomar el caso de una pequeña empresa de comidas conformada por pocos miembros para los cuales existe suficiente motivación de hacer rápido y bien las actividades asignadas pues el resultado se observara casi inmediatamente, con directas responsabilidades para el individuo; Ahora podemos citar el caso de gran escala del cuidado del medio ambiente

situación en la cual la dependencia de resultados gracias a los aportes individuales no es fácilmente evidente, también el compromiso entre el individuo y la colectividad es asumido sin la suficiente motivación.

Es así, que en los dilemas de gran escala la dependencia de resultados gracias a la cooperación individuo-grupo no es tan evidente motivo por el cual el individuo preferirá elegir sus opciones de un modo egoísta, en el caso del medio ambiente entonces deteriorando el mismo y desmejorando las posibilidades de sostenibilidad.

Cuando se sabe que muchos individuos actúan con reciprocidad en situaciones particulares, existe la ventaja de que cualquiera gane la reputación de ser confiable y se comporte con reciprocidad. En el núcleo de una explicación conductual de niveles de cooperación mayores a los previstos, en la mayoría de los dilemas sociales se trata de conectar entre "la confianza que los individuos tienen en los demás, la inversión que los demás hacen en reputaciones confiables, y la probabilidad de que los participantes usarán normas recíprocas.

A continuación se muestra una comparación de características para dilemas de pequeña escala y dilemas de gran escala:

TABLAI
Comparación Dilemas Sociales según la escala [1]

Tipo	Pequeña escala	Gran escala
Contexto	Campo y laboratorio	difícil de localizar
Tamaño grupo	menor 10	más de 10
Características grupo	homogéneas	heterogéneas
Magnitud retardo	pocos minutos	alta, días, meses o años.
Calidad realimentación	alta	baja
Modelo racionalidad	acotado	acotado
Encuentros	uno, finitos	infinitos
Comunicación	frente a frente	mediada medios masivos

IV. EL SOFTWARE LIBRE

El software libre nació de la mano del propio software en la década de los años 60[2]. Entonces las gigantescas maquinas a las que llamaban computadoras hacían uso de programas cuyo código fuente estaba a la vista de todos (los que querían verlo, por supuesto) y se podía distribuir libremente. Esto provocó que ya en esos tiempos, prehistóricos desde el punto de vista de la informática, existiera una pequeña comunidad de científicos y programadores que intercambiara código, a la vez que informes de errores e ideas. El software por entonces no era más que un valor añadido a las carísimas computadoras y se solía distribuir gratuitamente por los fabricantes.

La situación cambió radicalmente con el descenso del precio de las máquinas y sus componentes (el hardware) y la progresiva necesidad de un software más potente y con mayores funcionalidades. La ventaja competitiva que el intangible daba a las maquinas llego hasta el punto en el que incluso había gente que estaba dispuesta a pagar dinero por él. Esto que en sí no es necesariamente malo, provoco sin embargo un giro radical en la industria informática: las primeras compañías exclusivamente dedicadas a la creación de software aparecieron en el horizonte y se hicieron fuertes en el mercado. En aras de maximizar beneficios (económicos y estratégicos), una de sus tácticas habituales era limitar hasta más no poder lo que el usuario podía hacer con el software que creaban.

De repente, algo tan natural hasta pocas fechas antes como compartir un programa o su código se convirtió en una práctica deleznable y que atentaba no solo contra el creador del software, sino contra toda la industria del software y, por si acaso, también contra la sociedad y su bienestar.

No fue hasta mediados los años 80, cuando Richard Stallman formalizo las ideas básicas del movimiento del software libre que está revolucionando la industria del software. El software libre, tal y como lo conocemos hoy, dio sus primeros pasos con un manifiesto en favor de la libertad de expresión y un proyecto conocido hoy mundialmente, el proyecto GNU. Y con él, vio la luz probablemente una nueva forma de ver y entender el software y los bienes intangibles que se ha visto acelerada con la masiva implantación de Internet en las postrimerías del siglo XX y principios del actual.

V. PROGRAMAS PARA CONTROL Y SEGUIMIENTO DE VERSIONES

En vista de la dificultad para efectuar seguimientos y recolección de datos confiables y suficientemente globales de los casos representativos del software libre estudiados en esta investigación fue posible optar por la estrategia de hacer un estudio basado en datos históricos de producción de cada proyecto, el volumen de interacción de los miembros en aportaciones hacia el proyecto de desarrollo y versiones de desarrollo. Las primeras dos anteriores se pudieron encontrar fácilmente en el seguimiento histórico que hace la plataforma de control de versiones a cada proyecto de desarrollo.

Hablando un poco de estas plataformas se puede decir que en la actualidad existen variadas herramientas para desarrollo de software, ahora bien cuando los programas se tornan en un tamaño monstruoso inmanejable por mero ordenamiento para un administrador de proyectos, es entonces que es necesario recurrir a un software especializado en gestionar código fuente para la asimilación del mismo por varios o tal vez miles de usuarios que intentaran hacer modificaciones sobre varias partes del código fuente que este gestione, al dia de hoy el concepto de programa de gestión de versiones ha ido evolucionando hasta llegar al nuevo concepto de administrador de configuración de software (en inglés Software Configuration Management), ya que en realidad es necesario que un software de control de versiones configure tantas situaciones como el control de usuarios, transacciones para la sustitución de código, apertura de preguntas y errores, y más comúnmente las versiones de un archivo de

código fuente, las evoluciones de este archivo a través del tiempo, y el cruce de referencias con bibliotecas y similares.

Para este estudio nos interesó particularmente el uso que los usuarios le dan a este sistema ya que esto nos provee de una perspectiva directa y clara sobre el ritmo y tamaño de interacciones que tiene el usuario programador en el sistema que se usa para producir el producto, que el proyecto de software libre está creando. Cuestión que de otra manera seria prácticamente imposible obtener; los proyectos de software libre utilizan normalmente herramientas de control de versiones tales como CVS, Subversion, SourceSafe, ClearCase, Darcs, Bazaar, Plastic SCM, Git, Mercurial, Perforce

En esta investigación se usaron los datos históricos disponibles en la plataforma de controlo de versiones que ofrece sobre tecnología web el proyecto www.github.com, con las siguientes direcciones así:

Linux https://github.com/torvalds/linux
Freebsd https://github.com/freebsd/freebsd
Gentoo/kde https://github.com/gentoo/kde
Joomla-cms https://github.com/joomla/joomla-cms
PhpMyAdmin https://github.com/phpmyadmin/phpmyadmin
Ruby on Rails https://github.com/rails/rails

En la siguiente tabla se puede apreciar el nivel de reconocimiento de los casos de studio citados segun el volumen de desarrolladores

TABLA II

Proyectos de software libre y número de participantes oficiales

PROYECTO	Volumen de participación
Kernel GNU/LINUX	Más de 3700 desarrolladores, más de 6 millones de usuarios.
Berkeley Software Distribution (FreeBSD)	Auspiciado por más de 482 desarrolladores y grandes empresas de mercado del internet
K Desktop Environment (GENTOO / KDE)	65 colaboradores, Más de 60 traductores al español, más de mil en otros idiomas.
Joomla CMS	266 Desarrolladores, Usado en más de 112 idiomas
PhpMyAdmin	425 colaboradores
Ruby on Rails	2544 colaboradores, más de 3500

.

VI. RESULTADOS: IDENTIFICACIÓN COMPORTAMENTAL

Para entrar en el análisis de los comportamientos que la comunidad de desarrollo presenta en diferentes periodos de tiempo, teniendo en cuenta circunstancias propias de cada proyecto como lo son el número de miembros, la versión que realiza, las posibles versiones paralelas que está soportando, entre otros factores; es necesaria una especificación de comportamientos que haga enmarcarble cada caso representativo como válido para ser analizado mediante los esquemas teóricos de la teoría de cooperación aplicada a dilemas sociales de gran escala, de tal modo se puede tener un alto grado de confiabilidad en las propuestas de lineamientos para los proyectos de desarrollo de software libre, estos condicionamientos que cumple cada caso representativo, se listan a continuación como:

- *Realimentación y circularidad*
- *Iteración entre variables, Retardos y Percepción*
- *Representación comportamiento promedio*
- *Capacidad explicativa*
- *Representación proceso de toma de decisiones dinámicas*

En el proceso de observación de diferentes situaciones inherentes al software libre en cada proyecto de desarrollo, se pudo concluir que en su respectivo orden las dificultades ampliamente marcadas en los de utilización de mecanismos de cooperación en cada caso son:

- *Superación de condiciones de desconfianza*
- *Sostenibilidad de la cooperación*
- *Realimentación de información*
- *Cooperación como norma*
- *Enfrentar deserción*
- *Percepción de daño*

En respuesta a las causales anteriormente listadas, los lineamientos de cooperación propuestos aquí no pretenden ser estrategias agresivas de eliminación total de la falta de cooperación en los proyectos de desarrollo de software libre, sino más bien las pautas que produzcan en las políticas internas de los proyectos de desarrollo del software libre la posibilidad de disminuir a niveles poco dañinos la falta de cooperación por deserción, desconfianza entre otros causas; Estos lineamientos se relacionan inversamente a las causas mismas de fallos en los mecanismos de cooperación identificados en los proyectos, los mismos los podemos listar como:

- *Promover la llegada de nuevos miembros:* aunque los proyectos de software libre dependen de la rotación de desarrolladores(4) para mantener su ritmo de crecimiento y producción, no existen políticas formales de motivación hacia el desarrollador que recién comienza
- *Aprovechar las bifurcaciones:* Normalmente las bifurcaciones dentro de los proyectos de software libre son vistas como posibles peligros (5), pero si se mantiene una política de integración de versiones, las diferencias en profundidad de cada versión de desarrollo no será demasiado alejada de la rama principal y los grandes cambios serán siempre tomados poco a poco en cortos periodos de tiempo.

- *Implementar estructura de reconocimiento:* Los proyectos de software libre normalmente dependen de líderes naturales o de empresas que sean representantes de cada proyecto de desarrollo, razón por la cual no siempre los miembros desarrolladores simpatizantes se ven representados o beneficiados sin un reconocimiento en algún grado más directo, de tipo externo o interno(6) siendo mecanismos de reconocimiento interno una forma alcanzable de darle al miembro desarrollador un incentivo por su esfuerzo .

- *Cambiar no Desertar:* Los miembros desarrolladores desde los comienzos de problemas de privatización con el sistema operativo BSD optan en caso de desmotivación en el desarrollo del proyecto por simplemente abandonarlo e ir en otra dirección desconocida, esta propuesta de lineamiento busca que se implementen procedimientos de incorporación de desarrolladores dentro de otros campos del mismo proyecto(7) como por ejemplo desarrollo, pruebas y compatibilidad, medición de rendimiento, búsqueda de innovación, traducciones entre otros, en vista de que resulta más provechoso tener personas con experiencia anterior en el mismo proyecto de desarrollo(8).

- *Seguimiento de la calidad:* Aunque es difícil creer que con una ingeniería del software en vías de maduración, los proyectos de software libre no disponen de herramientas de seguimiento directo a la calidad(9) , cuestión que dificulta en términos de cortos periodos de tiempo a los proyectos de desarrollo, esto por la baja realimentación que se produce para la comunidad de desarrolladores, porque aunque los miembros desarrolladores son los primeros en probar el software producido es de notar que los usuarios finales producen un más alto sinnúmero de escenarios de uso que no son reportados hacia el proyecto de desarrollo de forma eficiente, sino más bien con altos periodos de retardo. Este lineamiento busca la introducción de procedimientos automatizados de medición de factores de calidad en las aplicaciones publicadas de los proyectos de software libre.

III. Conclusiones

En las observaciones realizadas se enmarcar tópicos esenciales de la teoría de la cooperación ha permitido reconocer en los casos de estudio dinámicas de cooperación a nivel interno de los proyectos de desarrollo de software libre que afectan directamente el comportamiento de crecimiento del recurso en los proyectos.

La realimentación de información se toma en los casos de estudio expuestos como la capacidad del desarrollador de ver reflejada en una versión de software publicada sus aportaciones durante el periodo de desarrollo, hecho que hace de una versión publicada el recurso del cual dispone una comunidad de desarrolladores y que para los casos de estudio expuestos se comprueba un comportamiento incremental por cada versión publicada, característica dependiente del volumen de aportaciones efectuado por los miembros durante los periodos relacionados a esta.

Si bien el retardo de la información hace que los desarrolladores que cooperan en un proyecto de software libre perciban sus aportaciones en el proyecto como una parte de la versión publicada, este proceso de asimilación de información no es igual para todos los miembros como incentivo motivador de para cooperar en el proyecto. De la misma forma la reciprocidad encontrada en los miembros a través del análisis de valles de aportaciones en momentos de la vida de un proyecto de desarrollo de software libre no es ni inmediatamente continua, ni proporcional a la ejecución de actividades cooperativas por parte del resto de la comunidad de desarrollo.

Finalmente fue apreciable que en proyectos de desarrollo de software libre el desarrollador de software atraviesa por situaciones problemáticas que hacen que iniciarse o mantenerse activo en un proyecto de desarrollo de software libre sea una decisión oscilante que se incline por el peso que agreguen variables de motivación tanto personales como grupales.

Reconocimientos

Agradecimientos a los autores de las obras referenciadas en esta investigación por la profundidad de los argumentos que sirven como herramienta para mejorar procesos de desarrollo para la mejora en la calidad de vida de la humanidad.

Referencias

[1] Parra V. Jorge Andrick, Evaluación de la Cooperación en Dilemas Sociales de Gran Escala, 2012
[2] Matellán O. Vicente, Compilación de ensayos sobre software libre,2004.
[3] Parra V. Jorge Andrick, Constructo para la evaluación de la cooperación en dilemas sociales de gran escala, 2010.
[4] Seoane Pascual Joaquín - Robles Gregorio, UOC Introducción al software libre. 2003
[5] Fogel Karl, Producing Open Source Software, How to Run a Successful Free Software Project, 2010.
[6] Vermeir Dirk, How Open is the Future?, 2005
[7] *Ostrom Elinor, El Gobierno De Los Bienes Comunes, 2000.*
[8] Moineau Laurent, Papathéodorou Aris Cooperación y producción inmaterial en el software libre, 2000.
[9] González B. Jesús M., Compilación de Ensayos sobre Software Libre, 2004.