Universidad de las Ciencias Informáticas

FACULTAD 10

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Título: Propuesta de una guía para el lanzamiento de proyectos de código abierto exitosos.

Autoras: Maura Oliva Acosta
Dayneris Prieto Colina

Tutora: Ing. Laya del Carmen Rabasa Frómeta

Ciudad de La Habana, 2009
"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos autoras únicas del presente Trabajo de Diploma y reconozco a la Facultad 10 de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2009.

_____ Firma del Autor
Maura Oliva Acosta

_____ Firma del Autor
Dayneris Prieto Colina

Dayneris Prieto Colina

_____
Firma del Tutor
Ing. Laya del Carmen

DATOS DE CONTACTO

"Frase".

AGRADECIMIENTOS

Dedicatorias

RESUMEN

El tema de esta investigación es proponer una guía estratégica con los aspectos a seguir por lo proyectos de código abierto que pretendan hacer su lanzamiento a la comunidad de forma exitosa. El estudio e implementación de esta guía brindaría una base a estos nuevos proyectos de código abierto evitando así que se olviden de definir y desarrollar elementos fundamentales como son, el manejo de la documentación, anuncio del proyecto así como el manejo de voluntarios. Se pretende establecer el marco teórico de la investigación, definir los aspectos técnicos que va a contener la guía y por último, valorar la propuesta mediante opiniones de un grupo de especialistas en el tema de la administración de proyectos. Como resultado de esta investigación se obtendría un guía estratégica para el lanzamiento de proyectos de código abierto, además la conformación de una serie de planillas destinadas a los proyectos para que estos no incurran en pasar por alto aspectos importantes a la hora de hacer el lanzamiento a la comunidad.

Palabras Clave

Código abierto

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El software se ha convertido en uno de los instrumentos más poderosos conocidos por el hombre, en estos momentos interviene prácticamente en todas las actividades directas e indirectas que realiza el ser humano. Dejar que éste sea controlado y restringido por personas solo interesados en su propio lucro supone un perjuicio irreparable para la sociedad. El software libre (en lo adelante SWL) que una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente brinda una importante oportunidad para tomar el control del destino.

El software que actualmente soporta el desarrollo de una economía más eficiente y productiva se encuentra monopolizado por parte de desarrolladores norteamericanos. Tal hecho facilita a los que se encuentran en dicha posición una elevación de los precios y la imposición de estándares propietarios y/o el pago por productos de dudosa calidad. Hoy en día el manejo de las tecnologías referentes al software es un aspecto estratégico para el desarrollo de cualquier institución o país, basta con saber que, es la herramienta con la cual las personas acceden al conocimiento, es usado en diferentes campos ya sea en los negocios, educación, actividades de producción y la prestación de salud.

Con el surgimiento de Internet el software extendió sus fronteras y sumó con esto miles de adeptos deseosos de colaborar en su producción. Las comunidades que lo desarrollan brindan una importante oportunidad para impulsar esta contribución. Las personas que comparten experiencias en ellas pueden trabajar en distintas áreas o enfocarse en alguna en específico, en estas comunidades se vive un espíritu cooperativo, se trabaja constantemente en la búsqueda del mejoramiento, en la difusión del software de código abierto y del conocimiento.

Nuestro país ha puesto sus ojos en el sistema operativo Linux como plataforma informática oficial por éste ser clasificado dentro de los estándares de SWL. Las ventajas que proporciona el software que se desarrolla bajo un entorno libre son disímiles, su bajo costo, seguridad, privacidad, calidad y actualización aportando aplicaciones más robustas que ayuden al desarrollo del país, evitando la dependencia de los sectores productivos industriales. En muchos de los casos la motivación inicial para elegir el SWL viene dada por la situación económica, provocando que no se puedan afrontar los costos de licenciamiento. En varias instituciones se pone en práctica el uso del SWL tanto en la parte administrativa como en la académica permitiendo aplicar diversas soluciones que antes eran

más difíciles de desarrollar. El uso de SWL en las universidades es mucho más que una manera de optimizar los recursos disponibles. Es una oportunidad para los estudiantes de participar en un esfuerzo de colectividad global y abandonar el rol de receptores pasivos de tecnología producida en los países desarrollados y comenzar a ser partícipes en su creación. Es en estos centros donde nace la necesidad de brindar a la comunidad universitaria un espacio para el trabajo colaborativo. La Universidad de las Ciencias Informáticas (en lo adelante UCI) no se quiere quedar detrás y se ha dado a la tarea de impulsar este desarrollo tecnológico.

Con el trabajo colaborativo de las comunidades de desarrollo se pretende que las personas colaboren entre sí y se retroalimenten de experiencias anteriores, es por esto que el centro ha considerado el desarrollo en comunidades como una vía más para impulsar los diversos proyectos productivos que en ella se implementan. El soporte tecnológico que en este momento respalda este desarrollo en la UCI es la herramienta GForge, utilizada para el trabajo colaborativo.

La experiencia adquirida en la facultad 10 de la universidad posibilita que actualmente cuente con alrededor de 60 proyectos comunitarios registrados en el sitio, esta facultad realiza su mayor esfuerzo para impulsar el desarrollo en comunidades mediante la base tecnológica de la plataforma anteriormente mencionada, este proceso es llevado a cabo paulatinamente debido a que la comunidad universitaria aún no cuenta con una amplia cultura sobre el trabajo en comunidades, frenando que se limite la participación de nuevos voluntarios. En la práctica educativa de la UCI se aprecian insuficiencias con el uso de información referente a los proyectos comunitarios, haciendo que a la mayoría de estos proyectos se les dificulte mucho lograr el éxito una vez salidos a la comunidad, hasta la consolidación de sus objetivos. Estas insuficiencias son la no realización de informes de avances periódicos, la falta de comunicación y coordinación para trabajar en equipo, poco uso de los canales de difusión, la definición de los objetivos y las tareas a cumplir por los integrantes de los equipos de desarrollo aspectos que se ven afectados en muchos de los proyectos comunitarios.

Se hace necesaria la definición de una guía estratégica para el lanzamiento de los proyectos de código abierto, la cual va a servir de base para todos los proyectos que pretendan ser lanzados de forma exitosa a la comunidad. En la guía estratégica se relacionarán los aspectos que deben ser cumplidos por los mismos a la hora de su apertura.

La situación anterior conduce a plantear el siguiente **problema**, los proyectos de código abierto no cuentan con una guía estratégica de lanzamiento a la hora de su salida a la comunidad.

La **pregunta científica** que se pretende responder para el desarrollo del tema es:

¿Cuáles son los aspectos claves que definen la guía estratégica para el lanzamiento exitoso de proyectos de código abierto a la comunidad?

Para hacer la propuesta de estos aspectos se hizo necesario el estudio de temas relacionados con el SWL y sus aplicaciones, denominaciones de las comunidades de desarrollo, además de la influencia que tenían los lanzamientos de los distintos proyectos nacionales e internacionales, de ahí se infiere el **objeto de estudio** los proyectos exitosos de código abierto comunitarios y el **campo de acción** guía estratégica para el lanzamiento de los proyectos de código abierto de la universidad a la comunidad.

Primeramente es necesario plantearse los objetivos para verificar en qué medida se cumplen de acuerdo a las acciones que se realizan, por tal motivo esta investigación se ha propuesto como **objetivo general** Diseñar una guía estratégica con los aspectos a seguir por los proyectos de código abierto que pretenden ser lanzados a la comunidad de forma exitosa.

Para cumplir con tal objetivo, se plantearon **objetivos específicos** como:

- Establecer el marco teórico de la Investigación.
- Definir los aspectos técnicos a cumplir por los proyectos de código abierto para su lanzamiento exitoso a la comunidad.
- Valorar la propuesta de desarrollo a partir de opiniones de un grupo de especialistas en cuanto al tema.

Con el objetivo de guiar, controlar y evaluar la investigación se definieron las siguientes **tareas**:

- Revisión de los aspectos bibliográficos relativos a comunidades de desarrollo y lanzamiento de proyectos de código abierto, establecer un diagnóstico de las tendencias actuales.

- ❖ Definición de los aspectos esenciales que conformarán la propuesta a cumplir por los proyectos.
- ❖ Evaluación del contenido de la información obtenida y realizar un estudio comparativo de los aspectos propuestos con los utilizados en los casos de estudio.
- ❖ Realización de encuestas a administradores de proyectos de código abierto de la universidad para valorar la propuesta.
- ❖ Elaboración de plantillas donde se recoja toda la información necesaria para el lanzamiento de los proyectos a la comunidad.

Se utilizaron varios **métodos** del nivel teórico para dar cumplimiento a las tareas propuestas tales como el **análisis - sintético** que permite estudiar el comportamiento de toda la información obtenida relacionada con la temática que se aborda, descomponer e integrar mentalmente y llegar a conclusiones parciales y generales sobre lo investigado. El **inductivo - deductivo** a la hora de conocer las características generales de los proyectos de código abierto para lo que se requiere hacer primeramente un estudio de una muestra de sujetos, mientras mayor sea el número de proyectos investigados más fácil será llegar a una generalización de las características más comunes de los proyectos. La **modelación** posibilita crear el modelos para investigar la realidad, constituyendo una reproducción simplificada de la misma que cumple la función heurística posibilitando el eslabón intermedio entre el sujeto y objeto de investigación a través del diseño de una guía estratégica constituyendo un elemento de peso en la solución de la contradicción actual y el deseado. El **histórico - lógico** a la hora de realizar el análisis de la evolución y avance de las comunidades virtuales de desarrollo y diferentes tipos de proyectos de código abierto. Cuando se desarrolló el objeto de estudio se partió del estudio de proyectos exitosos internacionalmente y se estudiaron los aspectos que contribuyeron a un lanzamiento exitoso. Todo esto se realizó con el objetivo de llegar a definir una guía estrategia más específica con los elementos necesarios para hacer lanzamiento exitoso. También se utilizaron métodos del nivel **Empírico**, entre ellos encuestas a los administradores de proyectos comunitarios con el objetivo de diagnosticar en qué grado los proyectos escogidos cumplían con cada punto de la guía estratégica planteada. Se utilizó el muestreo no probabilístico dentro de este se trabajó con el muestreo intencional porque fueron escogidos los proyectos que han permanecido más tiempo activo en cuanto al número de

descargas. Se tomó como población proyectos de la plataforma GForge con sus respectivos administradores y se les dio el cuestionario y a otros les fue enviado por correo electrónico. El tamaño de la muestra que con la que se trabajó fue determinado de forma cualitativa por lo que se analizaron las características y los objetivos de lo escogido. Estadísticamente se establecen límites porcentuales en la proporción que debe guardar la muestra en relación con el tamaño de la población; en término general, se considera que el límite mínimo de confiabilidad se sitúa en el 10% de la población. [1] La encuesta con un total de catorce preguntas fue respondida por el ciento por ciento de los encuestados. Esta fue utilizada mayormente como técnica de recopilación de información a la hora de realizar el estudio exhaustivo del campo de acción de la investigación así como para la valoración de la propuesta. Estas encuestas les fueron realizadas a líderes y administradores de proyectos comunitarios que se escogieron como caso de estudio de la plataforma GForge. La realización de entrevistas a los administradores de algunos de los proyectos comunitarios escogidos para valorar la propuesta en el capítulo tres. También se utilizaron métodos **matemáticos** y **estadísticos** como el análisis porcentual y la estadística descriptiva, para el procesamiento e interpretación de los datos empíricos obtenidos en tablas.

Resultados esperados

Con el logro de los objetivos propuestos se obtendría una guía estratégica con los aspectos a cumplir por los proyectos de código abierto que les servirá de base para realizar su lanzamiento. La cual contendrá aspectos fundamentales que definen las condiciones a cumplir por los proyectos para su lanzamiento a la comunidad. La definición de esta guía estratégica permite que los proyectos no olviden definir elementos esenciales a la hora de su apertura a la comunidad. Ejemplo de ello: la misión del proyecto, el estado de desarrollo, las formas de obtención del mismo, las características que debe tener la documentación asociada al proyecto. Todos estos aspectos van a contribuir al éxito del proyecto una vez lanzado a la comunidad aunque no lo garantiza. Estos aspectos se pueden encontrar en las cinco plantillas elaboradas por las investigadoras para la organización de los aspectos para la futura presentación ante los proyectos.

La guía obtenida puede ser estudiada por cualquier proyecto que se desee desarrollar de forma comunitaria, lo cual hace extensible la utilización de este trabajo en el entorno colaborativo de

desarrollo de la universidad y en este mismo esfuerzo a nivel de país cuando las condiciones de recursos permitan extender este tipo de desarrollo a la comunidad cubana de software libre.

Valor práctico

Se espera que con el cumplimiento de esta investigación, se cuente con una guía estratégica de lanzamiento donde se definan los aspectos o tareas fundamentales con los que deben contar los proyectos de código abierto a la hora de hacer un lanzamiento exitoso a la comunidad.

Estructura del informe

La presente investigación consta de Introducción, Capítulo 1, 2, 3, conclusiones parciales, conclusiones generales, recomendaciones, referencias bibliográficas además de la bibliografía utilizada durante el desarrollo del trabajo y por último los anexos que complementan el cuerpo del trabajo que conjuntamente con el glosario de términos, ayudan al entendimiento del mismo.

El primer capítulo cuenta con la fundamentación teórica del tema con la finalidad de comprender lo que se está investigando, se da una descripción detallada del objeto de estudio, además de relacionar en un epígrafe, la herramienta que es utilizada en la universidad para el trabajo colaborativo. Se presenta el estado del arte de los lanzamientos exitosos de proyectos de código abierto a la comunidad, además de mencionar elementos que los caracterizan.

En el segundo capítulo se profundiza en el análisis y desarrollo de los aspectos que conforman la guía estratégica propuesta.

En el tercer y último capítulo se analizan los resultados de la propuesta conformada en el capítulo dos y se procede a la valoración de la misma. Se presenta la propuesta a un grupo de proyectos escogidos, se analizan los resultados y se dan las valoraciones de cada aspecto por el cual se hace el análisis de la encuesta a los proyectos escogidos.

Capítulo I. Fundamentación Teórica

La política del SWL aboga por un modelo de desarrollo que gira en torno a la comunidad. Existen innumerables ventajas al desarrollar un proyecto aprovechando las capacidades de trabajo colaborativo y el conocimiento de personas con los mismos intereses que colaboran para lograr un objetivo común. Sin embargo este desarrollo es guiado por una serie de aspectos que son esenciales para garantizar que una vez salido a la comunidad dicho proyecto tenga una buena aceptación por parte de los usuarios y atraiga el mayor número posible de colaboradores.

En este capítulo se exponen aspectos relacionados con el SWL, las comunidades virtuales, las herramientas de trabajo colaborativo, los proyectos exitosos de código abierto y por último se exponen los aspectos necesarios para que, una vez nacido el proyecto, éste no muera. El seguimiento de estos aspectos no determina el éxito rotundo del producto, pero si influye considerablemente en la consolidación del mismo.

1.1.1. Software Libre

El SWL tuvo sus orígenes en pequeños grupos de activistas que querían pasar de un extremo a otro esta industria. Después de muchos años esta comunidad ha crecido rápidamente e incorporado no solo voluntarios en todo el mundo, sino ha atraído la atención y la cooperación de centenares de instituciones.

Inicialmente el SWL fue popular en los servidores y con el paso del tiempo más el trabajo de miles de voluntarios las piezas faltantes se completaron. Linux que antes estaba limitado a ser un sistema que no era visible a los usuarios finales ahora es un sistema que es usado por miles de personas en todo el mundo en sus computadoras personales, teléfonos entre otros. Ejemplo de sistemas construidos sobre Linux lo constituyen Google y Amazon.

El SWL es propiedad de todos, cada persona en el mundo tiene derecho a usarlo, copiarlo y modificarlo de la misma manera que los autores. El término libre no se refiere al precio, sino a la libertad, no existe contradicción alguna entre la venta de copias y el SWL. De hecho, la libertad para vender copias es crucial, las colecciones de SWL a la venta en formato de CD-ROM son muy importantes para la comunidad y venderlas es una forma de recaudar fondos para el desarrollo del mismo.

El SWL es considerado un legado de la humanidad que no tiene propietario, de la misma manera que las leyes básicas de la física o las matemáticas. No existe un monopolio y no es necesario pagar algún importe por su uso.

1.1.2. Comunidades virtuales de desarrollo

Las comunidades virtuales existen mucho antes del surgimiento de Internet. Estudiosos del tema han creado diversas clasificaciones principalmente en cuanto a si están basadas en espacios físicos o a los intereses de sus miembros, a la vez que se sub-clasifican de acuerdo a su orientación en sociales, profesionales y comerciales.

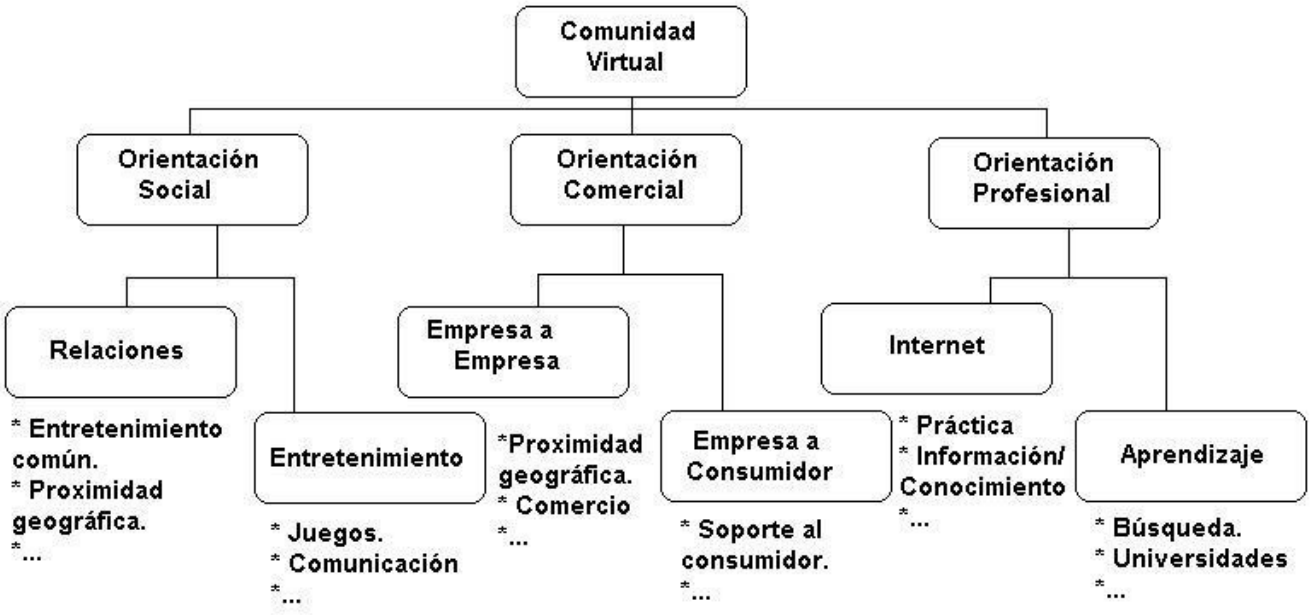

Figura 1: Caracterizando las Comunidades Virtuales

Dentro de las comunidades virtuales existentes en Internet, ocupan un lugar relevante las comunidades de desarrollo de software por cuanto han creado productos y servicios de altas prestaciones. [2] Para la creación de comunidades de desarrollo han de tenerse en cuenta tres aspectos imprescindibles que van a contribuir al desarrollo de proyectos con el modelo colaborativo. Estos aspectos son las herramientas de desarrollo, la divulgación de los proyectos y las guías de desarrollo.

Hoy en día las comunidades virtuales son una herramienta muy útil desde el punto de vista profesional, éstas, permiten a personas mejorar su dinámica de acuerdo al trabajo que desempeñan, las relaciones con las personas de su entorno, ya sea clientes o trabajadores en general a la vez que incrementan su eficiencia en los procedimientos. En cuanto a su función social, las comunidades se han convertido en un lugar en el que el individuo puede desarrollarse y relacionarse con los demás, actuando así como un instrumento de socialización y esparcimiento. Este tipo de comunidad queda definido por tres aspectos distintos:

- La comunidad virtual como un lugar: en el que los individuos pueden mantener relaciones de carácter social o económico.

- La comunidad virtual como un símbolo: ya que la comunidad virtual posee una dimensión simbólica. Los individuos tienden a sentirse simbólicamente unidos a la comunidad virtual, creándose una sensación de pertenencia.

- La comunidad virtual como virtual: las comunidades virtuales poseen rasgos comunes a las comunidades físicas, sin embargo el rasgo diferenciador de la comunidad virtual es que ésta se desarrolla, al menos parcialmente, en un lugar virtual, o en un lugar construido a partir de conexiones telemáticas. [3]

Las reglas o políticas de la comunidad deben funcionar correctamente. Esto se logra haciendo un buen manejo de voluntarios y haciendo que el liderazgo funcione adecuadamente para impulsar el trabajo colaborativo.

1.1.3. Tipos de comunidades de desarrollo

Las comunidades virtuales responden directamente al objetivo de sus miembros, a los requerimientos de afiliación, a los roles, a la historia compartida y a la construcción social de las reglas de participación. Por tanto no se puede dar una clasificación absoluta sobre los tipos de comunidades, ésta depende del propósito con que fue creada.

En el epígrafe anterior se hace mención a la clasificación más general de las comunidades virtuales, por lo que de acuerdo a sus características son las que agrupan mayor número de personas.

Orientación social

Este tipo de clasificación se puede subdividir en, “relaciones que se construyen” de acuerdo a afinidades de intereses, y de “entretenimiento”, las cuales son caracterizadas por el intercambio entre los miembros de la comunidad mediante el juego o alguna aventura.

El objetivo principal de la primera clasificación es establecer una relación entre los miembros de la comunidad virtual incitado sobre todo por algún interés personal común resultado de proximidad geográfica, de semejanza demográfica o simplemente, una manía común. El miembro individual está generalmente interesado en el grupo en conjunto es decir, el comprometimiento con el grupo es mucho más fuerte que con otros tipos de la comunidad.

Un ejemplo de esta categoría es la comunidad SeniorNet *(www.seniornet.org)*, fue establecida en 1986, con el objetivo de reunir personas de más de 50 años por medio de las computadoras e Internet. Y el tipo de reuniones que realizan son del tipo virtual y a veces de carácter físico.

Orientación comercial

Las comunidades orientadas comercialmente apuntan a tener ganancias o ganar una ventaja financiera. La decisión de estas comunidades en cuanto a si han sido exitosas o no, se ve reflejada en los llamados factores de beneficio, si la comunidad ha generado ventas o si los ahorros han sido utilizados en actividades específicas del negocio. Este tipo de comunidad es subdividido en dos categorías, “negocio entre empresas” y “negocio a consumidor”. Los negocios entre empresas son instalados ante todo para apoyar cadenas de suministro o colaboración entre empresas dentro de un área común geográfica. En la categoría negocio a consumidor son desarrolladas principalmente para apoyar un producto o la marca. Ellos son usados como instrumentos para adquirir y conservar a nuevos clientes, así reduciendo los costes de comercialización.

Un ejemplo típico de una comunidad comercial entre empresas es SupplyOn *(www.supplyon.com)* la cual está establecida para apoyar una cadena de suministros. Establecida en el año 2000 con el objetivo de juntar y/o reconciliar vendedores en la industria automotor para conjuntamente desarrollar usos para compra, ingeniería o logística.

Un ejemplo de una comunidad “negocio a consumidor” virtual es la llamada LEGO *(www.lego.com)* Establecida en el 2000 con el objetivo de proporcionar una plataforma para entusiastas LEGO para intercambiar información e ideas.

Orientación profesional

Las comunidades virtuales orientadas a la profesión se pueden subdividir en dos categorías: "aprendizaje", para adquirir conocimiento en un nuevo tema o el campo de interés. Y el "networks" (internet) experto que se caracteriza por ampliar, desarrollar, y documentar conocimiento existente. Las redes de aprendizaje pueden ser encontradas en las áreas de la educación y formación si el aprendizaje. Si se cambia de puesto al ambiente virtual y se requiere de la autentificación de los miembros al inscribirse a la comunidad. La característica fundamental de estas comunidades virtuales es que el interés común esta en aprender por sí mismo el tema que sea de interés personal.

Un ejemplo de red de aprendizaje es la red virtual de la educación dentro de la comunidad alemana de la universidad para los sistemas de información Winfoline *(winfo.uni-goettingen.de)* establecida en 1997 con el objetivo de fijar la red de la educación dentro de la comunidad alemana a cuatro universidades para los sistemas de información del negocio.

Los tipos de comunidades que se presentan a continuación, tienen, como primer objetivo, mostrar la diversidad y amplitud del fenómeno de Internet.

- Comunidades de debate y discusión en tiempo real.

Son las más visitadas. Se refieren a los canales de discusión a través de aplicaciones de chat o IRC. Estas aplicaciones permiten dialogar con todas las personas que en ese momento están en el mismo canal en tiempo real (just in time). La conversación es, generalmente escrita, aunque ya las aplicaciones permiten conversaciones con voz e imágenes si se dispone de un micrófono y una webcam. Algunos portales dan la posibilidad de abrir un canal privado de Chat donde invitar a otros participantes.

- Comunidades de socialización, información, discusión o juego.

Este tipo de comunidades está más estructurado y organizado. Las relaciones que se establecen son más duraderas y exigen una mayor implicación personal. Los que forman parte de ellas lo hacen conscientes de querer aprovechar las posibilidades de la Red para establecer relaciones y lazos más estrechos con personas de cualquier parte del mundo. También dentro de esta categoría

se pueden situar las redes cívicas, una interesante experiencia de comunidad virtual en un territorio determinado.

- Comunidades temáticas de investigación y producción.

Utilizando las aplicaciones típicas de Internet señaladas en el tipo anterior, estas comunidades se caracterizan por aglutinar a sus miembros alrededor de un tema concreto o una actividad de investigación. En estas comunidades no se limitan en darse a conocer utilizando la red como lugar de encuentro, de formación y de intercambio.

- Comunidades de organizaciones e instituciones.

Se puede incluir en este tipo las comunidades que surgen paralelas a instituciones gubernamentales, políticas o sociales. Internet sirve para crear redes internas al servicio de esa institución. Hasta ahora la mayor parte de las instituciones que mantienen sitios Web aprovechan una mínima parte de las posibilidades de Internet, función de marketing hacia fuera y comunicaciones internas. Pero las potencialidades del uso de las nuevas tecnologías en esta perspectiva van mucho más allá de eso y cambiaría la forma de ser de esas instituciones. [4]

Del tipo Aprendizaje Digital se pueden citar cuatro tipos de comunidades:

- De discurso

EL ser humano es una criatura social y puede hablar cara a cara sobre intereses comunes, pero también puede compartir estos intereses con otros semejantes más lejanos mediante los medios de comunicación. Las redes de ordenadores proporcionan numerosas y potentes herramientas para el desarrollo de este tipo de comunidades.

- De práctica

Cuando en la vida real alguien necesita aprender algo, normalmente no abandona su situación normal y dedica su esfuerzo en clases convencionales, sino que puede formar grupos de trabajo (comunidades de práctica), asigna roles, enseña y apoya a otros y desarrolla identidades que son definidas por los roles que desempeña en el apoyo al grupo. El aprendizaje resulta de forma natural al convertirse en un miembro participativo de una comunidad de práctica.

- De construcción de conocimiento

El objetivo de este tipo de comunidades es apoyar a los estudiantes a perseguir estratégica y activamente el aprendizaje como una meta (Esto es, aprendizaje intencional). Cuando los estudiantes poseen el conocimiento al mismo tiempo que el profesor o el libro, adquieren confianza para construir conocimiento, en lugar de recibirlo y reproducirlo solamente. La construcción del conocimiento se convierte en una actividad social, no una solitaria actividad de retención. La tecnología puede jugar un importante papel en las comunidades de construcción de conocimiento al proporcionar medios de almacenamiento, organización y reformulación de ideas aportadas por cada miembro de la comunidad.

- De aprendizaje

Las comunidades de aprendizaje surgen cuando los estudiantes comparten intereses comunes. Las TIC pueden contribuir a conectar alumnos de la misma clase o de alrededor del mundo, con el propósito de lograr objetivos comunes. [5]

Las comunidades que son de interés para esta investigación son las desarrolladoras de software las cuales se encuentran dentro de la clasificación de profesionales de las de investigación y producción. Esta clasificación se le atribuye a la universidad por contar con una comunidad de SWL desde el año 2006 apoyada por la Federación Estudiantil Universitaria, siendo su principal objetivo desarrollar proyectos de código abierto, y para lograrlo ha realizado acciones que lo han posibilitado. Dentro de estas acciones se encuentra la creación de un Portal de SWL, la instalación del sistema operativo Linux en todas las computadoras dedicadas a la docencia de una facultad, entre otras.

1.1.4. Groupware, categorías

El término de Groupware es designado al entorno donde todos los integrantes del proyecto trabajan, se ayudan y colaboran para la realización del mismo. Se refiere al software para el trabajo en grupo o trabajo colaborativo. Todo este software se basa en una correcta gestión del conocimiento y se necesita que sea accesible, rápido, se pueda corregir lo que está mal, y añadir nuevos elementos. El groupware es la integración de la filosofía de trabajo en grupo con las tecnologías de la información, está basado en los principios de cooperación, comunicación y coordinación.

El software para el trabajo en grupo mejora el rendimiento de todo el proceso productivo, su mayor aporte es hacer que las personas trabajen de forma compartida con una misma información y cooperen con el desarrollo del proyecto.

El groupware se puede dividir en tres categorías:

Herramientas de colaboración-comunicación

Se caracterizan por el envío de mensajes, archivos, datos o documentos entre personas los cual facilita el intercambio de información de forma asíncrona. Ejemplo:

- Correo electrónico.
- Correo de voz.
- Publicación en Web.

Herramientas de conferencias

Facilitan el intercambio de información de forma interactiva. Ejemplo:

- Conferencias de datos.
- Conferencias de voz.
- Sistemas para organizar reuniones.
- Salas de Chat o mensajería instantánea.

Herramientas de gestión colaborativa

Facilitan las actividades del grupo. Ejemplo:

- Calendarios electrónicos.
- Sistemas de gestión de proyectos.
- Sistemas de control de flujo de actividades.
- Sistemas de soporte a redes sociales.
- Sistemas de gestión del conocimiento. [6]

Uno de los elementos que sustenta el trabajo colaborativo lo constituyen las herramientas de desarrollo o herramientas colaborativas como se le suelen llamar a las que se les hará referencia en el siguiente epígrafe.

1.1.5. Herramientas de desarrollo. GForge

Son múltiples las herramientas libres y propietarias que hoy facilitan el trabajo en equipo. Además soportan toda la infraestructura facilitando la comunicación y colaboración entre los integrantes del grupo de desarrollo distribuidos en la red. Entre los más populares de todos los tiempos en el para el hospedaje de proyectos de código abierto se destacan SourceForge, GForge y más recientemente LaunchPad. Estas herramientas (nombradas canned hosting (enlatados) en inglés) agrupan en sí una serie de funcionalidades útiles para el desarrollo de proyectos colaborativamente. En la universidad, teniendo en cuenta la estabilidad del servicio así como la experiencia previa en la implantación en proyectos internacionales, se encuentra disponible un entorno colaborativo de desarrollo soportado por GForge.

GForge

GForge es un producto que integra un conjunto de herramientas aptas para el desarrollo de software de código abierto. Entre las herramientas cabe destacar foros, herramientas de gestión y monitorización de errores y tareas, listas de correo, repositorios de ficheros y herramientas de control de versiones. GForge es por tanto, un producto que proporciona una infraestructura básica sobre la que se apoya la construcción de comunidades de desarrollo de software de código abierto, que parte de la última versión liberada de Sourceforge. A continuación se presentan algunas de las funcionalidades con las que cuenta:

- Foros de discusión.
- Comunicación mediante listas de distribución de correo.
- Tratamiento de incidencias.
- Distribución de noticias significativas para el proyecto.
- Documentación compartida.
- Descargas de archivos (release de ficheros).

- Gestión de la planificación de tareas.
- Herramientas para crear y administrar repositorios utilizando CVS (Concurrent Version System) o Subversión.
- Publicación de recortes de código.
- Administración de Tareas.
- Administración de miembros, roles y referencias.
- Reportes de estadísticas. [7]

La plataforma GForge con que cuenta la Facultad 10 constituye un ambiente en el cual se hospedan proyectos de forma que el código, la documentación, los binarios, etc., son accesibles públicamente por todo el que desee verlos, y el público pueda contribuir con opiniones, detección de errores, ideas y sugerencias; además de ayudar a desarrollar el código, módulos, documentación y recursos para el software. Permite el control de versiones. Incluye un sitio Web por proyecto, y herramientas para la comunicación entre los miembros del equipo de desarrollo. Sus herramientas permiten a los miembros de un equipo de desarrollo una mejor organización del trabajo, y crear una base de conocimiento para futuros proyectos. Es considerado una herramienta muy poderosa para el desarrollo colaborativo de la comunidad del software.

GForge es una bifurcación del código fuente 2.61 de SourceForge, El proyecto de GForge fue formado y es mantenido por Tim Perdue. [8]

La herramienta utilizada por la facultad presenta algunas de las funcionalidades descritas a continuación:

- Administración de tareas.

Es similar al seguimiento de registros con algunas funcionalidades incrementadas, dentro de las que se encuentran: permite agregar tareas con fecha de comienzo y finalización, y estas pueden tener dependencias con otras tareas, porcentaje completado, entre otros aspectos. Cuando se le es asignada una tarea a un miembro del equipo de desarrollo, éste recibe un correo con la notificación y además el vínculo de donde está la orientación de la tarea.

- Distribución de noticias significativas para el proyecto.

Se distribuyen noticias referentes a algún acontecimiento importante para el proyecto que sirva de motivación tanto a usuarios como a desarrolladores.

- Publicación de recortes de código.

Permite a todos los usuarios interesados descargar el código del programa que desee, ya sea para estudio individual y aplicación posterior. O también hacer mejoras en el mismo contribuyendo con el trabajo colaborativo.

- Reportes de estadísticas.

Se reflejan la cantidad de proyectos registrados en la plataforma. De estos se pueden observar los que han tenido mayor actividad en la semana, los de mayor número de descargas. Además de informar cuales fueron los nuevos proyectos registrados.

- Administración de documentos.

GForge brinda una manera sencilla de publicar documentos en el sitio y tiene funcionalidades que muestran los documentos activos y pendientes.

- Seguimiento de errores.

GForge da la posibilidad de monitorear los errores detectados de forma automática. Cuando se activa el monitoreo, cada cambio en el error será enviado por e-mail, lo que posibilita hacer un seguimiento de las modificaciones en tiempo real.

- Seguimientos de registros.

Es un sistema genérico dónde se pueden almacenar ítems como errores, nuevos requerimientos, inclusión de parches, etc. Se puede utilizar para rastrear virtualmente cualquier clase de dato.

- Encuestas para usuarios y administradores.

Las encuestas permiten hacer preguntas a los desarrolladores y usuarios y ver resultados de forma automatizada.

- Administración de versiones de ficheros y repositorio de ficheros.

Tiene un módulo de integración con los diferentes sistemas de control de versiones, ejemplo: CVS, subversión, el cual es utilizado para subir archivos al sitio y ponerlos disponibles a los usuarios de manera fácil y eficiente.

1.1.6. Proyectos de código abierto exitosos internacionales

Los programas informáticos están sujetos a economías de redes. El importe de las aplicaciones se acrecienta cada vez más a medida que aumenta el número de beneficiarios, consumidores, interesados, usuarios en fin. Surgen mercados en los que una única empresa líder alcanza una predominancia abrumadora. Los consumidores pasan a depender de una sola tecnología que todos utilizan como consecuencia del alto costo de cambiar y aprender a usar productos alternativos.

A continuación se mencionan algunos de los proyectos exitosos de código abierto más conocidos en los que el equipo que los desarrolla es considerado una comunidad:

KDE *(www.kde.org)*, Gnome *(www.gnome.org)*, OpenOffice.org (*openoffice.org*), Debian *(www.debian.org)*, SourceForge *(www.sourceforge.net)*, Launchpad *(www.launchpad.org)*.

1.1.7. Características de los proyectos exitosos de código abierto internacionales

Existen varios tipos de proyectos de código abierto. Algunos son considerados herramientas para el soporte de estos proyectos como son los portales y otros sencillamente, aplicaciones. Estas aplicaciones pueden ser sistemas operativos, distribuciones, aplicación de escritorio, servidores Web y navegadores.

Es importante el conocimiento de su surgimiento, estado actual y la dirección que siguen para entender cómo funcionan y hacia dónde se dirige el SWL.

KDE

Tipo: Aplicación de escritorio.

Licencia (para aplicaciones) GPL, QPL, MIT, Artistic.

Licencia (para bibliotecas) LGPL, BSD, X11.

Herramientas de desarrollo: CVS, listas de correo, sitio web, sitio de noticias.

En 1996 se inicia el proyecto KDE con el objetivo de crear un entorno de escritorio gráfico para Unix. Dentro de este proyecto existen muchas aplicaciones, una de ellas es el Kmail, un completo

gestor de correo electrónico. Konkeror, siendo un navegador Web y a la vez un gestor de archivos. Koffice, que constituye un paquete ofimático de prestaciones básicas.

Aunque el desarrollo de KDE se realiza principalmente por voluntarios, existen diversas empresas como Mandrake, Suse (ahora parte de Novell) o TrollTech, que colaboran con recursos y desarrolladores a tiempo completo en el proyecto. Otra de las empresas que colabora es Xandros comercializando escritorios avanzados para entornos Unix.

Los lenguajes de programación utilizados en esta aplicación de escritorio son C++, C, Objetive C, Shell, Java y Perl. [9]

GNOME

Tipo: Aplicación de escritorio.

Licencia: GNU GPL y GNU LGPL.

Herramientas de desarrollo: CVS, listas de correo, sitio web, sitio de noticias.

Se lanza con el objetivo de crear un entorno de escritorio sencillo y moderno para Unix. En esos momentos la decisión de usar una librería propietaria para implementarlo enojó a sectores de la comunidad de SWL, debido a que el objetivo que perseguían muchos hackers, no estaba acorde con la situación que estaba afrontando el proyecto lo que se pretendía era hacer un sistema operativo completamente libre, y una parte tan importante del mismo no podía basarse en software privativo.

Después de muchos meses de esfuerzos, en marzo de 1999 en el marco de Linux World Expo en San José, se anunciaba GNOME 1.0. Consistía en una versión importante que daba fin a más de dos años de esfuerzos y que proporcionaba una plataforma para el desarrollo de aplicaciones que aprovecharan toda su funcionalidad. Desde entonces GNOME ha ido mejorando y creciendo a un ritmo imparable, y hoy en día se destaca por su alto grado de usabilidad.

GNOME tiene una fundación que coordina la liberación de nuevas versiones y determina qué proyectos forman parte del mismo. La fundación actúa como voz oficial ante los medios de comunicación y coordina la creación de materiales educativos y documentación destinada a guiar

usuarios con el objetivo de aprender a usar el entorno. Además, representa al proyecto en conferencias y contribuye a crear los estándares técnicos y especificaciones para el mismo.

Los lenguajes de programación utilizados en esta aplicación de escritorio son C, C++, Perl, Shell, Python, Lisp. [9]

OpenOffice.org

Tipo: Aplicación ofimática

Licencias: LGPL y SISSL

Herramienta de desarrollo: CVS, Listas de correo.

En 1999 Sun Microsystems adquirió la compañía alemana StarDivision. Su producto estrella era StarOffice, un paquete ofimático bastante maduro en aquella época, orientado al público tanto personal como profesional. Siguiendo los pasos de Netscape, Sun decidió liberar el código de StarOffice y crear una comunidad de SWL. Así nacía OpenOffice.org, dando el nombre a la comunidad que mantiene la versión libre del proyecto y a la propia comunidad. En poco tiempo, OpenOffice.org se convertía en la solución ofimática más utilizada dentro del mundo del SWL.

Dicha solución se encuentra disponible en más de 30 idiomas e incluye un procesador de textos, una hoja de cálculo, un programa de presentaciones y un programa para crear gráficos. Una de las características más importantes es que permite abrir casi perfectamente un documento con Microsoft Office, lo cual representa una gran ventaja para poder trabajar con usuarios de este paquete.

Otra de las ventajas es que sus formatos nativos de datos están basados en XML. Esto representa una gran flexibilidad para poder crear herramientas para procesar documentos o recuperarlos en otras aplicaciones. Los lenguajes de programación utilizados en esta aplicación ofimática son C++, Java, C, Pascal, Perl. [9]

Mozilla/Firefox

Tipo: Navegador web

Licencias: MPL/MGPL/GPL

Mozilla comenzó como un proyecto comercial porque el número de voluntarios que participaban al crearse este proyecto, no cumplía con las expectativas de los desarrolladores.

Para contrarrestar esta situación se mejoró la documentación, fueron escritos tutoriales, y se han refinado las herramientas de desarrollo y procesos, la participación se inició poco a poco en aumento.

Sólo más tarde hacia 1998 adoptó un enfoque de código abierto, con el fin de comprender el impacto de este cambio, es esencial analizar el alcance y la naturaleza de la participación externa en las contribuciones de código y la presentación de informes. Con el paso del tiempo se ha visto un aumento de las personas voluntarias para proporcionar aportaciones de código y esto se ve claramente en el avance que ha tenido el grupo de desarrollo del proyecto.

Mozilla tiene en la actualidad seis productos, cuenta con equipos de prueba que asumen la responsabilidad de probar diferentes partes o aspectos del producto, tales como el cumplimiento de las normas, el correo y cliente de noticias. Las inspecciones se hacen en dos fases: módulo de propietarios, que son los que hacen la revisión de un parche en el marco del módulo. Y un pequeño grupo designado con personal capacitado, que se encarga de la revisión de un parche para su interacción con la base de código en su totalidad.

El trabajo en el proyecto Mozilla no se limita a estos, sino que es mucho más diverso, se apoya en muchas tecnologías como son: Gecko y XULRunner. Además de incluir herramientas de desarrollo como CVS, Bugzilla, Camino, Firefox y Thunderbird.

GNU/LINUX (Sistema Operativo)

El Sistema Operativo Linux se ha convertido en el proyecto insignia de este movimiento de tal forma que muchos usuarios no iniciados, confunden el término Linux, con el de SWL. Linux nació de la mano de Linus Torvalds, en 1991 anunció en un foro de Internet la primera versión del núcleo de su sistema operativo. Torvalds había estado trabajando durante seis meses en crear un sistema operativo para sistemas Intel 386. A los pocos días de ser publicado numerosos hackers empezaron a contribuir con mejoras y arreglos. El núcleo que Torvalds había escrito era precisamente el corazón que le faltaba al sistema GNU que Richard Stallman estaba diseñando desde principios de los 80.

Así se creaban las primeras distribuciones Linux, que incluían el núcleo desarrollado por Torvalds más un conjunto de herramientas GNU. Hoy en día, existen distribuciones enfocadas a usos específicos como hacer de cortafuegos en una red o de servidores en Internet, o para ámbitos específicos como a educación, o de uso general.

Entre las distribuciones comerciales de Linux más conocidas se encuentran Red Hat, Mandrake Linux y Suse. Todas ellas están pensadas para el usuario final y son de amplio uso. Dentro de las distribuciones Linux merece una especial mención Debian. Esta distribución fue creada en agosto de 1993 con el objetivo de proporcionar una distribución totalmente libre. Actualmente, tiene más de 10.000 paquetes de software listos para instalar.

Linux no sólo ha innovado como software sino también en su modelo de producción. Torvalds impulsó desde el inicio un ritmo de liberación de versiones constante bajo el lema de liberar pronto y liberar frecuentemente, con este sistema Torvalds iba publicando versiones de Linux cada pocas semanas que imprimían un gran dinamismo al desarrollo del producto. Torvalds, que aún es el responsable del proyecto, ha sido capaz de gestionar las contribuciones al proyecto de una forma sensata, lo que le ha valido el apodo de dictador benevolente.

DEBIAN (Distribución)

Debian es un sistema operativo libre, utiliza el núcleo Linux (el corazón del sistema operativo), pero la mayor parte de las herramientas básicas vienen del Proyecto GNU; de ahí el nombre GNU/Linux. Comenzó en agosto de 1993 gracias a Ian Murdock, como una nueva distribución que se realizaría de forma abierta, en la línea del espíritu de Linux y GNU. Comenzó como un grupo de pocos y fuertemente unidos hackers de SWL, y gradualmente creció hasta convertirse en una comunidad grande y bien organizada de desarrolladores y usuarios.

Funciona en casi todos los ordenadores personales, incluyendo la mayoría de los modelos más antiguos. Cada nueva versión de Debian generalmente soporta un mayor número de arquitecturas de ordenadores.

Las razones por las cuales este producto tiene éxito entre las personas esta dado por varios aspectos.

Es mantenido por los mismos usuarios. Tiene un gran soporte en cuanto a las respuestas de correos. Contiene el mejor sistema de empaquetamiento a la hora de instalar software y gran variedad de software que permiten correr programas propietarios bajo GNU/Linux. Además de poseer una instalación sencilla mediante un CD. El código fuente se puede usar libremente para estudiarlo o para incorporarlo a un nuevo proyecto de SWL. También hay una buena cantidad de herramientas y código apropiado para el uso en proyectos propios.

APACHE (Servidor Web)

Cuando en 1994 se crea la empresa Netscape quedando atrás programas como el navegador Mosaic, creado por estudiantes de la Universidad de Illinois y que habían popularizado la Web desde esos momentos. Uno de estos programas quedó atrás, tal fue el caso del servidor Web NCSA httpd. El programa era libre y tenía muchos usuarios que lo seguían utilizando pero le hacía falta mejoras. Así se fue creando una colección de parches para añadir funcionalidad extra al código antiguo de httpd. Y fue entonces que hacia 1995 surgía el proyecto Apache, tomando como base estos parches y el trabajo del programa httpd de NCSA.

Apache es uno de los ejemplos de comunidades de desarrollo que han sido grandemente productivas coordinando más de 800 contribuidores voluntarios, de empresas y universidades. El proyecto cuenta con el Apache Group que está formado por desarrolladores que han colaborado durante un período largo de tiempo y que son escogidos por votación para formar parte del núcleo de personas que toman las decisiones.

UBUNTU (Distribución)

Cada lanzamiento o liberación de una versión de Ubuntu tiene mayor impacto y aceptación por los usuarios de la comunidad, cada liberación es mejor a la anterior, y nunca sus salidas demoran más de seis meses. Aunque no proporciona actualizaciones de seguridad y soporte profesional para todos los paquetes, sí selecciona un conjunto de paquetes para la creación de un sistema de escritorio sólido y completo brindando soporte para ellos. Ubuntu ofrece un componente o repositorio (conjunto de paquetes) “Univers”, donde los usuarios de esos sistemas pueden instalar la última versión de cualquier paquete que no esté en el conjunto de paquetes soportados.

La mayoría de los proyectos que se toman como exitosos tienen su propio sitio Web el cual está a disposición de todos los usuarios interesados, en estos sitios se encuentra información referente al proyecto y todo lo que acontece sobre el mismo ya sean noticias, foros de discusión, encuestas. Estos proyectos presentan un gran número de desarrolladores, permanecen más tiempo activo en cuanto al número de descargas, tienen amplia documentación para todo tipo de público y publican grandes cifras de noticias. La clave del éxito de estos proyectos está, en que desarrollaron una buena estrategia de lanzamiento caracterizándose por una buena publicidad, se utilizaron adecuadamente los canales de comunicación para un mejor intercambio entre los usuarios y desarrolladores, todo esto trajo consigo una mayor aceptación por el personal que interactúa con el proyecto.

1.1.8. Proyectos de código abierto exitosos en la UCI

El desarrollo de software de código abierto en la UCI se ha visto plagado de innumerables altas y bajas, cuando se organizó por primera vez la comunidad de SWL, no existía una cultura de GForge, no había portal. No se conocían las herramientas colaborativas, ni estaba extendido el uso de las mismas. Hoy en día se ve como esta cultura ha ido creciendo paulatinamente, se cuenta ya con el Polo de SWL, y se trabaja en más de 60 proyectos pertenecientes a la comunidad.

Según estudios realizados en la plataforma GForge en cuanto al tema, se observan los proyectos que han tenido un mayor éxito a la hora de su lanzamiento a la comunidad. El proyecto más descargado al término de esta investigación y con mayor actividad es el proyecto QEVEN.

QEVEN

Sus siglas significan Qt Easy Video Encoder (QEVEN) permite codificar fácilmente a varios formatos de vídeo.

Es considerado el primer proyecto exitoso en la comunidad de SWL de la UCI. Como todo proyecto nuevo en su desarrollo presenta aspectos positivos y aspectos criticables, tales como:

Cosas Buenas:

- Presenta una organización impresionante para personas que no se ven.
- Poseen precompilados como producto de salida.

- Tienen su propio estándar de código.
- Aprovechan el sistema de encuestas y noticias.

Críticas:

- Aún no utiliza el sistema de registros, o sea el envío de tickets por correo.
- Su documentación no es excesiva, carece de algunos elementos mínimos como una lista de requisitos o concepción del sistema para adicionar a otros miembros al desarrollo.

El éxito de este proyecto viene dado por varios aspectos que se manejan por parte de la dirección del mismo, tales como:

1. QEVEN tiene más noticias publicadas en torno a lo que desarrolla.
2. Es el segundo en actualizaciones en el SVN.
3. Está entre los que más ficheros pública. Específicamente da soporte a varias distribuciones y a Windows para su producto.
4. Es el segundo en organización de tareas.
5. Es el único que ha publicado al menos una encuesta.
6. Disponible a toda la comunidad de SWL de la UCI. [10]

1.1.9. Objeto de Estudio

1.1.10. Caracterización de los proyectos de código abierto

En opinión de las investigadoras de acuerdo a estudios realizados, se puede decir que el éxito de los proyectos de código abierto, esta dado por ciertos aspectos que son de importancia a la hora de su lanzamiento. Es de sumo valor tener una buena estrategia de lanzamiento donde se va a proporcionar una visión clara de las metas, necesidades y hasta las limitaciones del proyecto tanto en hardware como en software, también se necesitan definir los objetivos del proyecto y el alcance del mismo. Todo esto se debe realizar de forma tal que sea algo novedoso, para llamar la atención del público. Otro aspecto que atrae la atención de usuarios es el diseño de la interfaz que debe ser amigable y con facilidad de uso, debe hacerse la mayor divulgación posible del proyecto ya sea poner anuncios en portales, temas en foros y actividades en las comunidades de desarrollo, es

preciso contar con una buena documentación tanto para usuarios como para desarrolladores, un proyecto que no cuente con eso, es muy probable que no tenga seguidores y por consiguiente su éxito se verá afectado.

1.1.11. Caracterización de los proyectos de código abierto de la UCI

La UCI, universidad surgida al calor de la Batalla de Ideas está trabajando arduamente en el lanzamiento de diversos proyectos de código abierto a la comunidad. La facultad 10 de dicha universidad es la encargada de enfatizar en esta tarea, para lo cual cuenta con un Portal de SWL y la herramienta GForge que permite a los usuarios de la comunidad interactuar entre sí. Aunque todavía no se cuenta con una cultura de lo que es el trabajo colaborativo en las comunidades de desarrollo, se viene enfatizando en ello como una vía más para apoyar los diversos proyectos productivos de la facultad y en un futuro extenderlo para todo el país. Actualmente existen aproximadamente 60 proyectos comunitarios disponibles a todo aquel que esté interesado en cooperar, muchas veces los participantes en esta comunidad se ven limitados a una mayor integración a los proyectos pues les falta información relacionada al proyecto que les interesa ya sea documentación, publicación de noticias, encuestas a sus desarrolladores, reportes y estadísticas, debido a estas razones y estudios realizados a la comunidad de SWL de la universidad se observó que solo un pequeño por ciento de estos proyectos son exitosos y la gran mayoría fracasa porque no cumplen con sus objetivos iniciales o por ser proyectos que nunca se implementaron. Otro de los problemas típicos que se cometen son la falta de una administración eficiente y la no realización de informes de avance periódicos. Estas situaciones están dadas a que estos proyectos no implementan una eficiente estrategia de lanzamiento.

1.1.12. Elementos necesarios para un lanzamiento exitoso

Muchos de los proyectos que surgen actualmente no son fructíferos por diversos motivos. Algunos no exponen una idea clara de sus objetivos y otros porque sencillamente no se especifican las tareas a desarrollar. Por todo esto, es necesario que se elabore una guía para los colaboradores, que dependerá de las características del proyecto. Esta guía además debe tocar elementos del diseño, estándares y políticas a seguir por el proyecto.

De acuerdo a los resultados obtenidos de las revisiones bibliográficas realizadas referidas a proyectos de código abierto exitosos, se ha obtenido una primera guía estratégica con los puntos que tendrán que cumplir estos proyectos para hacer su lanzamiento a la comunidad.

A continuación se mencionan los elementos que se consideran necesarios a la hora del lanzamiento:

- Aspectos iniciales.
  - Panorama internacional.
  - Preparar al proyecto para la versión pública.
  - Selección de un buen nombre.
  - Misión.
  - Carácter del proyecto.
  - Lista de características y requerimientos.
  - Anuncio del proyecto.
  - Hospedaje.
  - Disponibilidad.
  - Presentación.
  - Estado de desarrollo
- Aspectos de Documentación.
  - Documentación.
    - Documentación para usuarios.
    - Guías para desarrolladores.
    - Documentación para desarrolladores.

- Empaquetamiento, actualizaciones y versionado.
  - Aspectos de Infraestructura tecnológica.
- Infraestructura tecnológica.
    - Sitios Web.
    - Wikis.
    - Chat en tiempo real.
    - Lista de correo.
    - Control de versiones.
    - Gestión de fallos.
    - Seguimiento de tareas.
  - Aspectos comunitarios.
- Código de conducta.
- Manejo de voluntarios
  - Aspectos complementarios
- Aspectos financieros.
- Aspectos legales.

La migración a SWL en Cuba supondrá uno de los cambios más importantes de la Revolución Informática. Como todo cambio, requiere que sea enfrentado con seriedad, con una visión estratégica y un alto nivel de preparación que permitan mitigar los inconvenientes que esta situación trae consigo.

El uso del SWL en la sociedad no sólo es recomendable por el hecho económico de favorecer la competencia en el sector, reducir los costes y de limitar la dependencia tecnológica, también por el hecho de brindar mayor seguridad, mayor fiabilidad y mayores posibilidades de evolución que sus contrapartidas comerciales.

Prepararse para promover el desarrollo de software de código abierto en la comunidad cubana una vez que estén creadas las condiciones es de vital importancia y una contribución inestimable en el proceso de migración y soporte a plataformas de código abierto que lleva el país.

Conclusiones

La idea del uso de una guía estratégica para el lanzamiento de los proyectos de código abierto, conlleva al estudio de un grupo temas relacionados con los procesos de desarrollo, ya sea en la universidad, como en el resto de la isla. El hecho de que se tenga en consideración la elección de cada aspecto que conformará la guía estratégica, es de vital importancia porque cada uno de ellos será aplicado con la finalidad de que documente y contribuya al éxito de cualquier proyecto de código abierto que guíe su lanzamiento por ella.

Hasta este momento quedaron definidos los aspectos que las investigadoras suponen son de relevancia a la hora de lanzar un proyecto a la comunidad. Se hace necesaria la revisión de cada uno y la realización de un análisis para definir cómo va a ser su implantación en cada proyecto de acuerdo a las características presentes en ellos.

Capítulo II Diseño de la propuesta

Para trabajar con éxito en el SWL hay que entender por qué hay personas que lo eligen, y si lo hacen, cuentan con la posibilidad de si están en un proyecto y no les gusta, poder recurrir a otro y así, hasta que encuentre un lugar donde se sienta a gusto y sea útil. Muchos de los participantes nunca se encuentran físicamente, y lo que hacen simplemente es, utilizar una parte de su tiempo cada vez que se sientan motivados para trabajar en el proyecto. Es muy fácil que un proyecto pierda un voluntario potencial en muy poco tiempo de haberlo encontrado. Se debe trabajar en lo que el proyecto necesita tener para ganar prestigio entre los distintos tipos de usuarios que interactúan con el proyecto y para ganar voluntarios cada vez más.

En este capítulo se entra en detalles de la propuesta obtenida en el capítulo anterior. Se explican cada uno de los aspectos y en algunos casos se da una recomendación que en qué se deben fijar los administradores a la hora del lanzamiento y que el proyecto, tenga un impacto satisfactorio en la comunidad.

Una vez leído este capítulo el lector contará con una serie de elementos no sólo para evitar los errores comunes en el desarrollo de programas de código abierto, sino también para manejar el crecimiento y el mantenimiento de un proyecto exitoso. Un aspecto importante para impulsar un proyecto de fuente abierta es trabajar en armonía con otros proyectos relacionados entre sí. Cada proyecto exitoso contribuye al bienestar de todas las personas que colaboran con el SWL.

Cada uno de los siguientes puntos, describen un aspecto importante de iniciar un nuevo proyecto. Están presentados casi en el orden que las investigadoras suponen son fáciles de entender para una persona que posea conocimientos básicos del tema de administración de proyectos de código abierto. Incluso pueden ser tratados como una lista de tareas. Cuando se inicie un proyecto, debe asegurarse de revisar la lista y de que cada uno de los elementos sean cubiertos, o al menos asegurar cierta flexibilidad con las posibles consecuencias que traería dejar alguno aparte.

2.1.1. Aspectos iniciales

2.1.2. Panorama internacional

Antes de proponerse un proyecto de código abierto primeramente hay que revisar varios aspectos importantes como el panorama internacional, investigar si ya existe algún proyecto de este tipo.

Se debe investigar primeramente si existe un proyecto que hace lo mismo que lo que se propone lograr con el suyo. La posibilidad de que el problema que se quiere resolver este siendo llevado a cabo por alguien más o haya otra persona interesada en resolver un problema como este o similar son muy buenas y de gran ayuda. Si existe alguien resolviéndolo y lo ha hecho bajo una licencia libre entonces no será necesario comenzar desde cero. Existen excepciones en caso de que desee iniciar un proyecto como experiencia educativa, el código pre-existente no es de ayuda o quizás el proyecto que se desea iniciar es muy especializado y se sabe que no existe la posibilidad de que alguien más lo haya hecho ya. Pero generalmente, no hay necesidad en no investigar porque las ganancias pueden ser grandiosas.

Es importante hacer una revisión exhaustiva de todos los posibles sitios y plataformas que brindan información relacionada a los proyectos de código abierto. Incluso si no se encuentra exactamente lo que se busca, podría encontrar algo parecido, a lo que tiene más sentido unirse a ese proyecto y añadir funcionalidades en lugar de empezar desde cero.

2.1.3. Preparar al proyecto para la versión pública

Lo más difícil acerca de lanzar un proyecto de código abierto es transformar una visión privada a una pública. La organización que desea llevar a cabo dicho proyecto quizás sepa exactamente lo que quiere lograr con este, pero quizás se le haga un poco más difícil expresar ese objetivo de manera comprensiva al resto de la comunidad. De hecho, es esencial tomarse el tiempo necesario para hacerlo. Los fundadores deben decidir sobre qué desea realmente el proyecto, esto implica decidir sus limitaciones, o sea lo que no podrá hacer así como lo que debe cumplir, para esto es necesario escribir una declaración de objetivos. Ésta parte no suele ser usualmente difícil, aunque puede revelar afirmaciones y desacuerdos sobre la naturaleza del proyecto.

El próximo paso es empaquetar el proyecto para el consumo público. Lo que lo hace laborioso es principalmente organizar y documentar lo que ya todo el mundo conoce, o sea todos aquellos involucrados en el proyecto hasta entonces. Así que, para las personas trabajando ya, no existen beneficios inmediatos. Estos no necesitan de un documento que resuma el proyecto ni de un documento de diseño o manual de usuario. No necesitan de un árbol de código cuidadosamente ordenado conforme a los estándares informales, ampliamente utilizados para las distribuciones de fuentes. De cualquier forma como esté ordenado el código fuente estará bien, porque ya estarán

acostumbrados de todas formas, y si el código funciona, saben cómo usarlo. Un aspecto que no es muy relevante es si la arquitectura del proyecto se encuentra sin documentar, los desarrolladores ya están familiarizados con lo que deben hacer.

En cambio, los recién llegados, necesitan de todas estas cosas. Afortunadamente, no las necesitan todas a la vez. No es necesario proporcionar todos los recursos posibles antes de tomar un proyecto público. Quizás en un mundo perfecto, todo nuevo proyecto de código abierto empezaría su vida con un riguroso documento de diseño, un manual de usuario completo marcando especialmente las características planeadas pero que aun no han sido implementadas, código empaquetado y portable, capaz de ejecutarse en cualquier plataforma y así sucesivamente. En realidad, cuidar de todos estos detalles consumiría demasiado tiempo, y de todas maneras, es trabajo con el que podrían ayudar los voluntarios una vez que el proyecto esté en marcha.

Por otro lado, lo que sí es necesario, es que se realice una inversión apropiada en la presentación, de forma que un recién llegado pueda superar el obstáculo inicial de no estar familiarizado con el proyecto. [11]

2.1.4. Selección de un buen nombre

Para entender este aspecto se debe colocar en la posición de alguien que acaba de escuchar acerca de su proyecto, quizás por alguien que casualmente tropezó con éste mientras buscaba por alguna aplicación para resolver un problema. Lo primero que encontraran será el nombre del proyecto.

Un nombre genial no hará que automáticamente el proyecto tenga éxito, y un nombre malo no significa que éste acabado, en realidad un mal nombre probablemente podría hacer eso, se empezará asumiendo que nadie está activamente intentando hacer que su proyecto falle. De todos modos, un mal nombre puede desacelerar la adopción del programa porque la gente no se lo tome seriamente o porque simplemente les cueste recordarlos.

Un buen nombre:

- Da cierta idea de lo que el proyecto hace, o al menos está relacionado de una manera obvia, como si alguien conoce el nombre y sabe lo que hace, después lo recordaran rápidamente.

❖ Es fácil de recordar. En estos momentos el inglés se ha convertido en el lenguaje por defecto de Internet: nombres que son juego de palabras dependientes en la pronunciación de ingleses nativos serán opacos para muchos lectores no nativos en inglés. Si el juego de palabras que se utilice es particularmente llamativo y memorable, quizás sí valga la pena. Sólo cabe recordar que en todo el mundo hay diferentes lenguas y pensamientos de las personas, se debe buscar un nombre que sea el más aceptable para la mayoría de las personas que pudieran oírlo.

❖ No tiene el mismo nombre que otro proyecto y no infringe ninguna marca comercial. Esto es por buenos modales, y tener un buen sentido legal. No se desea crear confusiones de identidad. Se hace bastante difícil mantenerse al día con todo lo que hay disponible en la red, sin tener diferentes cosas con el mismo nombre.

❖ Está disponible como un nombre de dominio .com, .net, y .org. Hay que escoger alguno, probablemente .org, para promocionarse como el sitio oficial del proyecto. Los otros dos deben reenviar allí simplemente para evitar que terceras partes creen una confusión de identidad sobre el nombre del proyecto. Incluso si piensa en hospedar el proyecto en otro sitio puede registrar los dominios específicos del proyecto y direccionarlos al sitio del hospedaje. Ayuda mucho a los usuarios tener que recordar sólo un URL. [11]

2.1.5. Misión

Una vez que se ha encontrado el sitio del proyecto, lo siguiente que las personas hacen es buscar por una descripción rápida, una declaración de objetivos, para poder decidir (en menos de 30 segundos) si están o no interesados en aprender más. Esto debe estar en un lugar prioritario en la página principal, preferiblemente justo debajo del nombre del proyecto. La declaración de objetivos debe ser concreta, limitada y sobre todo, corta. Expresa de forma clara el objetivo general del proyecto de forma que una vez leída el lector pueda captar rápidamente la esencia de lo que se quiere conseguir con el proyecto.

2.1.6. Carácter del proyecto

Aquellos que sigan interesados después de leer la declaración de objetivos querrán más detalles, quizás un poco de documentación para usuarios o desarrolladores, y eventualmente querrán descargar algo. Pero antes que nada, necesitaran estar seguros de que es de código abierto.

La página principal debe poner claramente y sin ambigüedades que el proyecto es de código abierto. Esto puede parecer obvio, pero unos cuantos proyectos se olvidan de esto. Se debe poner en el sitio del proyecto el carácter del proyecto, además de la licencia bajo la cual de distribuye la aplicación. Estas constituyen piezas cruciales de información que a veces son aisladas a la página de descarga o a la página de los desarrolladores o a algún otro lugar el cual requería más de un enlace para llegar.

No se deben cometer los errores de omisión por parte del equipo de desarrollo de estos puntos esenciales. Una omisión como ésta puede hacer que se pierdan muchos desarrolladores y usuarios potenciales. Declarar desde el principio, justo debajo de la declaración de objetivos, que el proyecto es de código abierto, y mostrar la licencia exacta.

2.1.7. Lista de características y requerimientos

Debería haber una breve lista de las características que el software soporta, si se da el caso de que aun no ha sido completado, se puede listar de todas formas, pero señalando *planeado* o *en progreso* y el tipo de entorno necesario para ejecutar la aplicación. Hay que pensar en ésta lista como algo que se le daría a alguien que requiere un resumen del programa.

Los requerimientos del programa están dados por las plataformas de hardware que soporta, sistemas operativos para el cual bajo el cual puede correr, las prestaciones que la máquina debe tener, como son, el espacio en disco, la memoria disponible, entre otras.

La lista de características y requerimientos daría detalles que permitirían esclarecer el alcance de la misión del proyecto.

Con ésta información, los lectores podrán rápidamente tener una idea de si este programa tiene alguna esperanza de trabajar para ellos, y también pueden considerar involucrarse como desarrolladores.

2.1.8. Anuncio del proyecto

EL anuncio del proyecto es un aspecto muy importante porque es la manera con que se va a presentar el mismo al público. Cuando el proyecto se encuentra en una fase robusta, presentable, es cuando se debe proceder a la apertura del mismo, donde se tienen definidos bien los objetivos que persigue el mismo y a la vez a qué tipo de público desea atraer. Este proceso es sencillo, se

debe revisar que el sitio donde se va a realizar el lanzamiento sea uno que las personas los visiten a menudo y con el objetivo de encontrar proyectos nuevos. Además de las publicaciones que servirán para atraer a nuevos usuarios, especificando lo novedoso del proyecto, las características, los requerimientos, etc., además de expresar el aporte intelectual del mismo o la importancia de la aplicación generada.

2.1.9. Hospedaje

Este aspecto se refiere al lugar donde va a estar alojado el proyecto. Para gestionar el hospedaje de proyectos de este tipo, un punto que debe tenerse en cuenta es que: el sitio de hospedaje debe ser el mismo que donde se encuentra el proyecto.

Este punto es importante porque si el sitio de hospedaje no es el mismo que el del proyecto, se deben tener diferentes versiones de los ficheros generados de manera que aparezcan en el sitio web. Estos ficheros se refieren a la versión HTML pública que se genera cada vez que el fichero maestro FAQ (Preguntas Más Frecuentes) es modificado los cuales deben encontrarse en el sitio web.

Otra de las consecuencias es que no se tiene el control absoluto de la presentación deseada. Cuando se le realizan cambios a la página principal del proyecto, estos cambios no pasan siempre a la página del sitio de hospedaje.

El hospedaje de un proyecto está estrechamente relacionado con el tipo de infraestructura tecnológica que presente el mismo, por lo que se debe tener en cuenta una serie de funcionalidades con las cuales debe contar un sitio que alberga proyectos de este tipo. Esta infraestructura se explica más adelante en este capítulo.

Sólo el equipo de desarrollo puede decidir acerca de cuál sitio de hospedaje es el mejor para el proyecto. Si se decide utilizar alguno externo, se recomienda dejar abierta la posibilidad de cambiar a un servidor propio más adelante utilizando nombres de dominio personalizados para la página principal del proyecto. Se puede remitir la URL al sitio hospedado o tener una página totalmente personalizada detrás de la URL pública y llevar a los usuarios al sitio hospedado para funcionalidades más sofisticadas. Se deben organizar las cosas de manera que si decide cambiar de solución para el hospedaje, la dirección del proyecto no deba ser modificada.

2.1.10. Disponibilidad del software

EL software debe poder ser descargable como código fuente, en formatos estándares, los paquetes binarios (ejecutables) no son necesarios, a menos que el programa tenga requerimientos muy complicados para su compilado o dependencias que hagan hacerlo funcionar y sea muy laborioso para la mayoría de las personas.

El mecanismo de distribución debe de ser lo más conveniente, estándar y sencillo posible, un programa debe ser conforme a métodos de compilación e instalación estándar; entre más se desvíe de estos estándares, mayor será la cantidad de usuarios y desarrolladores potenciales que se den por vencidos y abandonen el proyecto confundidos.

Esto parece obvio, pero muchos proyectos no se molestan en estandarizar sus procedimientos de instalación hasta mucho después, diciéndose a sí mismos que esto lo pueden hacer en cualquier momento. De lo que no se dan cuenta es de que al dejar de lado el trabajo aburrido de terminar los procedimientos de compilado e instalación, en realidad están ralentizando todo porque desalientan a los programadores que de otra manera habrían contribuido al código. Más dañino aun, no saben que están perdiendo a todos esos desarrolladores, porque el proceso es una acumulación de eventos que no suceden, alguien que visita un sitio web, descarga el programa, intenta compilarlo, y falla, deja de intentarlo y lo abandona. Nadie en el proyecto se dará cuenta que el interés y la buena voluntad de alguien ha sido silenciosamente malgastada.

Las tareas aburridas con un alto beneficio siempre deben ser hechas al principio y disminuyendo de manera significativa las barreras de entrada a un proyecto, utilizando buenos paquetes brindan altos beneficios.

Cuando se lanza un paquete descargable, es vital que se le dé un número de versión único a éste lanzamiento, de manera que la gente pueda comparar dos versiones cualesquiera diferentes y saber cual reemplaza a cual.

2.1.11. Presentación del proyecto

Si el proyecto implica una interfaz gráfica para el usuario o si produce una salida gráfica o distintiva, habrá que poner algunos ejemplos en el sitio web del proyecto. En el caso de las interfaces, esto significa capturas. Para salidas, pueden ser capturas o sólo ficheros. Ambos dotan al usuario de gratificación instantánea, una sola captura puede ser más convincente que párrafos de texto

descriptivo y cháchara de listas de correo, porque una captura es la prueba indiscutible de que el programa funciona. Puede que tenga fallos, quizás sea difícil de instalar o que la documentación esté incompleta, pero esa captura sigue siendo la prueba de que con el esfuerzo necesario, se puede hacer funcionar.

Existen muchas otras cosas que se pueden poner en el sitio web del proyecto, si se tiene el tiempo, o si por alguna razón u otra son especialmente apropiadas: página de noticias, historia, enlaces relacionados, función de búsqueda, enlace para donaciones, etc. Ninguno de estos es necesario al principio, pero hay que tenerlos en mente para el futuro.

2.1.12. Estado de desarrollo

Las personas siempre quieren saber cómo va un proyecto. Para proyectos nuevos, desean saber la separación entre las promesas del proyecto y la realidad del momento. Para proyectos maduros, desean saber cuan activamente es mantenido, cuán seguido sacan nuevas versiones, la facilidad para reportar fallos, etc.

Para responder a estas dudas, se debe suministrar una página que muestre el estado del desarrollo, listando los objetivos a corto plazo del proyecto y las necesidades por ejemplo, la búsqueda de nuevos desarrolladores en un tema en particular. Ésta página también puede dar una historia de versiones anteriores, con listas de las características, de manera que los visitantes obtengan una idea de cómo el proyecto define su progreso y de cuan rápidamente se hacen avances de acuerdo a esas definiciones.

No hay que asustarse por parecer no estar preparado y no caer en la tentación de inflar el estado del desarrollo. Todos saben que el software evoluciona por etapas; no hay que avergonzarse en decir que alguna versión del software presenta fallos, y que es responsabilidad del desarrollador su uso. Este lenguaje no asustará el tipo de desarrolladores que son necesarios en esta etapa. En cuanto a los usuarios, una de las peores cosas que un proyecto puede hacer es atraer usuarios antes de que el software esté listo para estos. Una reputación por inestabilidad y fallos es muy difícil de hacer desaparecer una vez adquirida. La paciencia da sus frutos a largo plazo; siempre es mejor que el software sea más estable de lo que espera el usuario ya que las sorpresas gratas producen el mejor boca a boca.

2.1.13. Aspectos de Documentación

2.1.14. Documentación

La documentación es esencial. Debe haber algo para que las personas lean, aunque sea algo rudimentario e incompleto. Conseguir una declaración de objetivos y una lista de requerimientos, escoger una licencia, resumir el estado de desarrollo son tareas relativamente pequeñas que pueden ser completadas y a las que usualmente no es necesario volver una vez terminadas. La documentación, por otra parte, nunca está terminada realmente, lo cual puede que sea una de las razones por las cuales se retrase su inicio.

En caso de que el tiempo del que se dispone solo alcance para escribir documentación para una audiencia, que sea para los usuarios. Toda la documentación para los usuarios es, en efecto, documentación para desarrolladores también. Cualquier programador que vaya a trabajar en un proyecto necesita estar familiarizado con su uso. Luego, cuando se vea a los programadores haciendo las mismas preguntas una y otra vez, habrá que tomarse el tiempo de escribir algunos documentos aparte sólo para estos.

Algunos proyectos utilizan wikis para su documentación inicial o incluso para su documentación principal, esto es efectivo si y sólo si, el wiki es editado activamente por algunas personas que se ponen de acuerdo en cómo la documentación debe ser organizada. En la universidad las wikis no son lo suficientemente explotadas por parte de los integrantes de los proyecto y es por eso que a veces se ve limitada la comunicación y el intercambio de información entre los distintos usuarios que interactúan en el mismo.

Se recomienda a los proyectos que tengan un equipo encargado de la documentación que sea el responsable de redactar, editar y actualizarla. Se debe sacar una versión cada vez que salga una nueva liberación del producto.

2.1.15. Documentación para usuarios

Lo más importante de la documentación para un usuario inicial es lo más básico: cómo configurar la aplicación, una introducción de cómo funciona y quizás algunas guías para realizar las tareas más comunes. Pero a la vez son estas cosas las más conocidas por aquellos que escriben la documentación, así que resulta difícil para estos ver las cosas desde el punto de vista de los lectores, dificultando listar los pasos que parecen tan obvios que no merecen especial atención.

No existe una solución mágica para éste problema. Alguien debe sentarse y escribir todo esto para luego presentárselo a un usuario nuevo tipo y probar la calidad. Hay que utilizar un formato simple y fácil de modificar como HTML, texto plano, Tex o alguna variante de XML, algo que sea conveniente para mejoras rápidas, ligeras e imprevisibles. Esto no es sólo para eliminar cualquier trabajo innecesario a los escritores originales realizar cambios incrementales, sino también para quienes se unan al proyecto después y desean trabajar en la documentación. [9]

Una manera de asegurarse de que la documentación básica inicial se hace, es limitando su alcance. Al menos de ésta manera no parecerá que se está escribiendo una tarea sin fin. Una buena regla es seguir unos criterios mínimos:

- Avisar al lector claramente el nivel técnico que se espera que tenga.

- Describir clara y extensivamente cómo configurar el programa y en alguna parte al inicio de la documentación, comunicarle al usuario cómo ejecutar algún tipo de prueba de diagnóstico o un simple comando para confirmar que todo funciona correctamente. La documentación inicial es a veces más importante que la documentación de uso. Mientras mayor sea el esfuerzo invertido en instalar y tener funcionando la aplicación, mayor será la persistencia en descubrir funcionalidades avanzadas o no documentadas.

- Dar un ejemplo estilo tutorial de cómo realizar alguna tarea común. Obviamente, muchos ejemplos para muchas tareas sería mejor, pero si el tiempo es limitado, es mejor escoger una tarea en específico y llevar al usuario de la mano paso a paso. Una vez que se ve que la aplicación puede ser utilizada, se empezará a explorar qué más es lo que puede hacer y si se tiene suerte empezar a documentarlo ellos mismos. Lo que lleva al siguiente punto.

- Indicar las áreas donde se sabe que la documentación es incompleta. Al mostrar a los lectores que se es consciente de las deficiencias, es necesario alinearse con su punto de vista. La empatía les da confianza en que no van a tener que luchar para convencer al proyecto de su importancia. Estas indicaciones no necesitan representar promesa alguna de completar los espacios en blanco en una fecha en particular, es igualmente legítimo tratarlas como requisitos abiertos para ayudantes voluntarios.

Ese criterio es de una especial importancia, y puede ser aplicado al proyecto entero, no sólo a la documentación. Una gestión exacta de las deficiencias conocidas es la norma en el mundo del código abierto. No se debe exagerar en las faltas del proyecto, solo identificarlas escrupulosa y desapasionadamente cuando sea necesario (sea en la documentación, en la base de datos de fallos o en discusiones en la lista de correos). Nadie verá esto como derrotismo por parte del proyecto, ni como una responsabilidad explícita. Ya que cualquiera que utilice la aplicación descubrirá sus deficiencias por sí mismos, es mejor que estén psicológicamente preparados, entonces parece que el proyecto tiene un sólido conocimiento acerca de cómo va progresando.

Los proyectos deben incluir en la documentación para usuarios documentos como,

- Notas sobre las versiones del proyecto.

Son documentos de texto con alguna información relacionada con el proyecto que pueda ser de interés para el usuario.

- Guías sobre la instalación del proyecto.

Constituyen manuales de cómo instalar el proyecto de acuerdo a la versión que se desee usar y las características del hardware que requiere.

- Guía para usuarios.

Es la que contiene los pasos más elementales que necesitan conocer los usuarios en una aplicación, ejemplo de estos pasos son: navegar por algún ordenador, leer y enviar mensajes en los foros o correo electrónico.

- Guías de seguridad.

Son los manuales que incluyen las prácticas relacionadas con el trabajo en servidores remotos y locales en contra de la intrusión, la explotación y la actividad maliciosa. También debe relacionar las herramientas que intervienen en la creación de un entorno seguro en el proyecto.

- Preguntas Más Frecuentes (FAQ).

Se recomienda que todo proyecto tenga un documento con las preguntas que sean de interés común para los usuarios que entran al sitio del proyecto o al sitio donde este hospedado.

2.1.16. Guías para los desarrolladores

Si alguien desea contribuir con el proyecto, buscará alguna guía de desarrollo. Estas guías van a ser más sociales que técnicas, explican como los desarrolladores interactúan con el proyecto y con los usuarios, además de incluir cómo hacer las cosas. Los elementos básicos de una guía de desarrollo son:

- Enlaces a los foros para la interacción de los desarrolladores.
- Instrucciones en cómo reportar fallos y enviar parches.
- Alguna indicación de cómo el desarrollo es usualmente llevado a cabo.
- Políticas de cómo descargar.
- Políticas de cómo instalar (descripción detallada).

Algunas de las guías que deben tener los desarrolladores son:

- Arquitectura del proyecto.

Es la que explica los objetivos del proyecto, de que tipo es y todo lo que le interese al equipo de dirección reflejar en esa guía.

- Manual de estilo del código.

Debe ser seguido por todos los desarrolladores porque es el que especifica el lenguaje que se debe utilizar en el proyecto y además cómo debe ser la estructuración del código.

- Manual de estilo de la interfaz.

Debe reflejar las características de la interfaz para que la misma, sea coherente para todas las funcionalidades que se pretendan cumplir en el proyecto.

- Manual de CVS para los desarrolladores.

Esta guía debe dar una explicación del cómo se debe trabajar con el controlador de versiones que utilice el proyecto, cómo subir un documento, cómo trabajar con el código, entre otros.

2.1.17. Documentación para desarrolladores

La documentación para los desarrolladores es escrita para ayudar a los programadores a entender el código y puedan arreglarlo o extenderlo. Esto es algo diferente a las guías de desarrollo discutidas anteriormente. Las guías de desarrollo les dicen a los programadores como deben desenvolverse entre ellos, en cambio la documentación les dice como deben desenvolverse con el código en sí mismo. Por conveniencia las dos vienen juntas en un sólo documento.

A pesar de que la documentación para los desarrolladores puede ser de mucha ayuda, no existe ninguna razón para retrasar un lanzamiento por hacerla. Es suficiente para empezar que los autores originales estén disponibles (y dispuestos) a responder a preguntas sobre el código. De hecho, tener que responder la misma pregunta varias veces es una motivación muy común para escribir dicha documentación. Pero antes de que sea escrita, determinados contribuyentes serán capaces de desenvolverse con el código ya que la fuerza que hace que las personas utilicen su tiempo en leer el código base es que éste código les resulta útil. Si las personas tienen fe en ello, ninguna cantidad de documentación hará que vengan o los mantendrá.

2.1.18. Empaquetamiento, actualizaciones y versionado

Este es uno de los aspectos más importantes cuando el proyecto debe tomar una posición seria ante los usuarios. Esta no es tarea de una persona en específico, sino, se recomienda hasta tres personas trabajar en ella. El versionado una vez que se establezca, debe ser especificado por escrito y va a ser el utilizado para las liberaciones del producto. Aunque si va a usar como controlador de versiones el SVN, éste mismo le proporciona un número único de versión que identifica un estado común de todos los archivos del repositorio en cierto punto de tiempo.

2.1.19. Aspectos de infraestructura tecnológica

La infraestructura tecnológica de un proyecto está conformada por un conjunto de tecnologías que contribuyen con la captura e integración de la información. Algunas de estas son las referidas a continuación.

2.1.20. Sitios Web

Principalmente, conducto de información centralizado en un sentido del proyecto para el público. El sitio web puede también servir como una interfaz para otras herramientas del proyecto. La principal función de un sitio web es ofrecer una visión general clara y unir las distintas herramientas que le

dan soporte al proyecto, como son, sistema control de versiones, gestión de fallos, seguimiento de tareas, etc. [11]

2.1.21. Wikis

Una Wiki es una herramienta que posibilita la creación y desarrollo de sitios web de manera colaborativa, en los visitantes editar o extender su contenido de forma fácil exponiendo sus ideas y criterios acerca de diferentes autores. El término “wiki” (una palabra Hawaiana que significa “rápido” o “súper-rápido”) también es usado para la aplicación que permite este tipo de edición. Las wikis se pueden ver como una integración entre los servicios de IRC Chat con las páginas web, estos no trabajan en tiempo real así que dan la posibilidad de reflexionar y pulir sus contribuciones, pero son muy sencillos de utilizar, facilitando aun más la edición de una página web.

Dado que son una tecnología relativamente nueva y los proyectos aún experimentan con las diferentes maneras de utilizarlos, en este epígrafe sólo se ofrecerán algunas precauciones porque resulta más fácil analizar los usos equivocados de las wikis en lugar de analizar sus éxitos.

Si el equipo de desarrollo decide ofrecer un wiki, se debe poner un gran esfuerzo en tener una organización clara de las páginas y un diseño visual atractivo, de manera que los visitantes instintivamente sepan como incluir sus contribuciones. Igual de importante, hay que publicar estos estándares en el mismo wiki, de manera que los usuarios tengan un lugar a donde ir en busca orientación. Muy a menudo, los administradores de las wikis caen en la trampa de creer que porque presentan un gran número de visitantes se añade individualmente contenido de alta calidad al sitio, el resultado de todo esto debe ser también de la más alta calidad y esto no es cómo funcionan los sitios web. Cada página individual o párrafo puede que sea bueno al ser considerado individualmente, pero no será tan bueno si está encuadrado de forma desorganizada y confusa. Demasiadas veces, las wikis sufren de:

- Falta de principios de navegación: Un sitio web bien organizado hace que los visitantes se sientan como si supieran donde se encuentran en todo momento. Por ejemplo, si las páginas están bien diseñadas, la gente puede percibir las diferencias entre una región con la tabla de contenidos y otra con el contenido. Los contribuyentes del wiki respetarán tales diferencias también si y solo si las diferencias están presentes.

❖ Información duplicada: Frecuentemente las wikis acaban con diferentes páginas con información similar, porque los contribuyentes individuales no se han dado cuenta de la duplicidad. Esto puede ser una consecuencia de la falta de principios de navegación mencionados antes, en que los usuarios puede que no encuentren contenido duplicado si este no se encuentra donde esperaban encontrarlo.

❖ Audiencia objetivo inconsistente: Hasta cierto punto este problema es inevitable cuando existen tantos autores, pero puede ser ralentizado si existen guías escritas acerca de cómo crear nuevo contenido. También ayuda editar nuevas contribuciones al principio dando un ejemplo a seguir de manera que los estándares se vayan asentando. [11]

La solución más común para todos estos problemas es el mismo, tener estándares editoriales y mostrarlos no sólo publicándolos sino que editando páginas y uniéndose a estos. En general, las wikis ampliarán cualquier fallo en el material original, ya que los contribuyentes imitarán cualquier patrón que vean. No sólo hay que configurar el wiki y esperar que todo funcione a la perfección. Se debe preparar con contenido bien escrito, de manera que los colaboradores tengan una plantilla que seguir. Además de tener asociado un foro de comentarios que se añaden al final del documento y que el autor original o alguien que tenga el papel de editor pueda añadirlos al final del documento original con el fin de enriquecer el tema comentado.

2.1.22. Canales de comunicación

Usualmente los visitantes desean saber cómo pueden contactar con las personas detrás del proyecto. Hay que suministrar direcciones de listas de correo, salas de chat, canales en IRC y cualquier otro foro donde aquellos involucrados puedan ser contactados. Se debe dejar claro que los autores del proyecto están suscritos a estas listas, de manera que las personas vean una forma de dar apoyo a los desarrolladores. La presencia de estos en las listas no implica obligación alguna de responder a todas las preguntas que se formulan o de implementar todas las peticiones. A la larga, muchos de los usuarios probablemente ni siquiera se unan a los foros de todas maneras, pero estarán conformes con saber que podrían si fuese necesario.

En las primeras etapas de cualquier proyecto, no existe la necesidad de que haya una diferenciación entre los foros de los usuarios y los de los desarrolladores. Es mejor tener a todos los involucrados del proyecto hablando en conjunto en una misma sala. La distinción entre usuario y

desarrollador será muchas veces borrosa, hasta tal punto que la distinción no se puede hacer y la proporción entre programadores y usuarios usualmente es mayor al principio que al final, no se puede asumir que todo aquel que utilice el programa sea un programador que quiere modificarlo, sí se puede asumir que al menos está interesado en seguir las discusiones sobre el desarrollo y en obtener una visión de la dirección del proyecto.

A continuación se analizan algunos canales de comunicación más importantes.

2.1.23. Chat en tiempo real

Muchos proyectos ofrecen salas de chat, foros donde los usuarios y desarrolladores pueden hacerse preguntas y obtener respuestas instantáneas. Mientras que se puede llevar un servidor de IRC para el sitio web, lo primero que hay que hacer es decidir un nombre para el canal. La opción más obvia es utilizar el nombre del proyecto. Hay que dar publicidad a la disponibilidad del canal en el sitio web del proyecto, de manera que un visitante con una duda pueda verlo rápidamente.

Algunos proyectos tienen varios canales, uno para cada tema. Por ejemplo, un canal para problemas de instalación, otro para dudas sobre su uso, otro para charlas sobre el desarrollo, etc. Cuando el proyecto es joven, sólo debe haber un canal en el que todos hablan juntos. Luego, mientras el proyecto vaya creciendo, la separación de canales será necesaria.

2.1.24. Listas de correo

Las listas de correo son un elemento clave en la comunicación del proyecto. Si algún usuario es expuesto a algún foro aparte de las páginas web, probablemente sea a la lista de correos del proyecto. Pero antes de trabajar con las listas en sí mismas, se debe tomar experiencia con la interfaz esto es, el mecanismo por el cual se pueden unir a la lista. Esto nos brinda la regla número uno de las listas de correo:

Es recomendable usar un software para configurar las listas de correo y no hacerlo manualmente.

Este aspecto puede parecer demasiado difícil al principio. Manejar listas pequeñas de pocos integrantes manualmente puede ser tentador al principio, sólo hay que montar una lista de suscripción que te reenvía todo y cuando alguien envía un mensaje, se agrega (o elimina) su dirección de correo en algún tipo de fichero de texto que almacena todas las direcciones de la lista.

Un software para el manejo de listas de correo usualmente ofrece las siguientes características:

- Suscripción a través de correos o basada en web

Cuando un usuario se suscribe a la lista, debería recibir una respuesta de bienvenida sin demora, explicándole como seguir interactuando con el software y (más importante) como eliminar la suscripción. Esta respuesta automática puede ser modificada para contener información específica del proyecto.

- Suscripción al modo de resúmenes o al modo de mensaje por mensaje

En modo resumen, el suscriptor recibe el correo conteniendo toda la información de la actividad de la lista en ese día. Para los que desean seguir la lista indirectamente, es preferible que usen el modo resumen. Porque les permite revisar todos los temas a la vez y evitar las distracciones de los correos que llegan en momentos al azar. [11]

El objetivo de todo esto es sencillamente enfatizar que la administración de las listas de correo es un problema complejo sobre el cual se ha pensado mucho y que está casi resuelto. Ciertamente no es necesario convertirse en un experto, pero hay que resaltar que siempre hay lugar para el aprendizaje y que la administración de las listas ocupara algo de atención de vez en cuando durante la duración del proyecto.

2.1.25. Control de versiones

Un sistema de control de versiones (o sistema de control de revisiones) es una combinación de tecnologías y prácticas para seguir y controlar los cambios realizados en los ficheros del proyecto, en particular en el código fuente, en la documentación y en las páginas web. Si nunca antes se ha utilizado un control de versiones, lo primero que hay que hacer es conseguir a alguien que sí lo haya hecho y hacer que se una al proyecto. Hoy en día todo el mundo espera que al menos el código fuente del proyecto esté bajo un control de versiones y probablemente no se tomen el proyecto seriamente si no se utiliza este sistema con un mínimo de competencia.

La razón por la cual el control de versiones es universal es porque ayuda virtualmente en todos los aspectos al dirigir un proyecto: comunicación entre los desarrolladores, manejo de los lanzamientos, administración de fallos, estabilidad entre el código y los esfuerzos de desarrollo experimental y atribución y autorización en los cambios de los desarrolladores. El sistema de control de versiones permite a una fuerza coordinadora central abarcar todas estas áreas. El núcleo del sistema es la

gestión de cambios: identificar cada cambio a los ficheros del proyecto, anotar cada cambio con meta-data como la fecha y el autor de la modificación y disponer esta información para quien sea y como sea. Es un mecanismo de comunicación donde el cambio es la unidad básica de información. Es muy importante concentrarse en la selección y configuración de un sistema de control de versiones de forma que se fomente un desarrollo cooperativo. [11]

2.1.26. Herramientas controladoras de versiones

Existen diversas herramientas controladores de versiones tales como: Mercurial [12], GIT [13], Arch, Codeville *(codeville.org)*, CVSNT *(csvnt.org)*, Concurrent Versions System (CVS) y Subversion (*subversion.tigris.org*), siendo las dos últimas las más populares.

- CVS

CVS ha estado durante mucho tiempo, y muchos desarrolladores están ya familiarizados con él. Fue el primer sistema de control de versiones de código abierto con acceso a redes de área amplia para desarrolladores y el primero que ofreció verificaciones anónimas de sólo lectura, los que dieron a los desarrolladores una manera fácil de implicarse en los proyectos.

Este es el más antiguo de los dos con las siguientes características: CVS sólo versiona ficheros, no directorios; ofrece ramificaciones, etiquetado, y un buen rendimiento en la parte del cliente, pero no maneja muy bien ficheros grandes ni ficheros binarios.

Desventajas:

No ofrece facilidad para hacer referencia a cambios en múltiples ficheros, no permite renombrar o copiar ficheros dentro del sistema de control siendo tedioso reorganizar el código después de iniciar el proyecto, el soporte para las uniones es algo pobre, no trabaja muy bien con ficheros muy grandes o con ficheros binarios y algunas operaciones son lentas cuando muchos ficheros están involucrados. Tampoco soporta cambios atómicos.

- Subversion

Subversion fue escrito ante todo para reemplazar a CVS es decir, para acceder al control de versiones aproximadamente de la misma manera que CVS lo hace, pero sin los problemas o falta de utilidades que más frecuentemente molestan a los usuarios de CVS. Uno de los objetivos de

Subversion es encontrar la transición a Subversion relativamente suave para las personas que ya están acostumbradas a utilizar CVS.

2.1.27. Gestión de fallos

Este es uno de los puntos más fuertes en el modelo de SWL porque los usuarios que pertenecen a la comunidad que publican los errores encontrados, se deben que son tenidos en cuenta ya sea soluciones o informes solamente. Debe implantarse de forma sencilla que le permita a los desarrolladores obtener información suficiente, de forma sistemática con todos los detalles necesarios como son, problema que ocurre, nivel de importancia y posible solución. Y esta información deberá guardarse en una base de datos para que sea consultada si se desea conocer los detalles de los reportes hechos. Es utilizado no solo para fallos, sino también lanzamientos, tareas, etc.

Se pueden implementar vía correo electrónico, utilizando una dirección válida para que no ocurra el ruido. O mediante un sitio web, pero siempre se debe asociar una dirección de correo electrónico para la interacción con la base de datos que nos informa acerca de los errores con más precisión.

2.1.28. Seguimiento de tareas

Las herramientas para el seguimiento de tareas dependen de las características del proyecto que va a ser lanzado. Estos sistemas no tienen que ser muy sofisticados porque muchos de los colaboradores son voluntarios y no se les puede poner un esquema muy rígido en cuanto a la asignación de tareas, sino más bien, se deben publicar en un sitio para que cuando alguien lo visite, pueda ver las características de las mismas, ya sea el nivel de importancia, el plazo que se le asigna para su cumplimiento y las personas que están trabajando en ellas.

El sistema más sofisticado que se está utilizando para esto es Issuezilla, una evolución propietaria del sistema de gestión de errores libre vía web BugZilla. [9]

2.1.29. Aspectos comunitarios

2.1.30. Código de conducta

Es necesario establecer cómo será el comportamiento de los integrantes del proyecto. Esto está muy relacionado con el objetivo del proyecto, las características de la licencia que se está utilizando y además el manejo de voluntarios, al cual se le va a ser referencia en este epígrafe. Uno de los

códigos de conducta más usados es el escrito bajo la licencia Creative Commons Attribution-Share Alike 3.0, utilizado por la comunidad Ubuntu.

Este código de conducta se refiere al comportamiento que deben tener los miembros de la Comunidad del proyecto en cualquier foro, lista de correo, wiki, sitio web, canal IRC, reunión pública o correspondencia privada. El Consejo de la Comunidad del proyecto debe velar por el comportamiento de los miembros de la comunidad.

Ser considerado. Tu trabajo será utilizado por otras personas, y que a su vez dependerá de la labor de los demás. Cualquier decisión que usted tome afectará a usuarios y colegas, y esperamos que usted tome en cuenta esas consecuencias.

Ser respetuoso. La comunidad del proyecto y sus miembros deben tratarse entre sí con respeto. Todo el mundo puede hacer una valiosa contribución al proyecto. Es posible que no siempre estén de acuerdo, pero el desacuerdo no es una excusa para el mal comportamiento y los males modales. Es posible que poca experiencia dé cierta frustración de vez en cuando, pero no se puede permitir que la frustración se convierta en un ataque personal. Es importante recordar que una comunidad donde los contribuyentes se sienten incómodos o amenazados no es productiva. Esperamos que los miembros de la comunidad sean respetuosos cuando se trata de otros colaboradores, así como con personas fuera del proyecto, y con los usuarios de este.

Ser colaborador del proyecto. La colaboración y trabajo conjunto, la colaboración reduce la redundancia del trabajo realizado en el mundo del software de código abierto, y mejora la calidad del software producido. Debe tratar de colaborar con otros mantenedores del proyecto, así como con la comunidad antes de que esté interesado en el trabajo que se hace. Su trabajo debe ser hecho con transparencia y los parches del proyecto deben ser devueltos a la comunidad cuando se hagan, no sólo cuando la distribución de prensa. Debe mantener al mundo exterior informado de su trabajo, y publicarlo de forma tal que permita a personas ajenas discutir y contribuir con su esfuerzo.

Cuando no esté de acuerdo, consulte a otros. Los desacuerdos, tanto políticos como técnicos, pasan todo el tiempo y la comunidad del proyecto no va a ser la excepción. El objetivo no es evitar los desacuerdos o diferencias de opinión, sino resolverlo de forma constructiva. Debe dirigirse a la

comunidad y al proceso comunitario para buscar consejo y para resolver los desacuerdos. Los jefes de equipo, pueden ayudarle a averiguar la dirección que será más aceptable.

Cuando no esté seguro, pida ayuda. Nadie sabe todo, y nadie está obligado a ser perfecto en la comunidad del proyecto. Hacer preguntas evita muchos problemas en el camino, y así se alienta. Lo que se pide debe ser rápido y útil. Sin embargo, al hacer una pregunta, se debe tener cuidado de hacerlo en un foro apropiado y solicitudes de ayuda en una lista de correo. *(www.ubuntu.es.org)*

2.1.31. Manejo de voluntarios

Lo importante en este aspecto es pensar que cada nueva persona que llega al proyecto es un voluntario potencial. La incorporación de voluntarios es de especial atención a medida que crece el proyecto por las distintas tareas que hace falta realizar en el proyecto. Sencillas tareas como programación, estructuración y actualización de la información, hasta desarrollos más complejos. Lo que debe dejarse claro es que cada tarea por muy simple que sea, sirve de aporte al desarrollo del proyecto. [11]

Se deben crear normas para el comportamiento colectivo tales que el prestigio sea adquirido y mantenido, a través de acciones que ayuden a la consecución de las metas del grupo. Esto se hace mediante la delegación de tareas que es muy necesaria porque le da a cada cual el rol a desempeñar en el proyecto de acuerdo a sus habilidades anteriormente verificadas. De esa forma también el trabajo de organiza.

La especificación de los roles en la comunidad deben ser descritos en algún documento. Estos roles pueden ser: desarrolladores, diseñadores, evaluadores de documentación, traductores, entre otros. La asignación de un rol u otro dependerá de las habilidades que tenga el voluntario.

2.1.32. Aspectos Complementarios

2.1.33. Aspectos legales

En el desarrollo de proyectos, el tema de la legalidad involucra aspectos como son las licencias, derechos de copia, derechos de autor, secretos industriales y patentes.

En las sesiones siguientes se puntualizarán algunas de estas categorías.

2.1.34. Licencias

La licencia de software es el instrumento legal a través del cual el autor, titular o proveedor (licenciante) establece las condiciones y los requisitos generales bajo los que se otorga el software al usuario (licenciatario). En fin, la licencia no es más que un contrato entre el autor y los usuarios que estipula lo que los últimos pueden hacer con su obra, uso, redistribución, modificación y en las condiciones que se debe realizar.

En cuanto al tema de las licencias que van a ser implantadas en los proyectos se deben analizar las libres existentes, la protección de la marca, la compatibilidad con otras licencias además de la integridad que supone, estos aspectos más el propósito del proyecto, tributarán a decidir el tipo de licencia a usar.

Antes de entrar en detalle con los ejemplos de licencias, un aspecto muy importante que de ser de conocimiento para aquel produzca software y tenga que definir el aspecto legal es que: cada nueva versión de un programa se considera como una nueva obra. El autor tiene, otra vez, plena potestad para hacer con su obra lo que le apetezca, incluso distribuirla con términos y condiciones totalmente diferentes (o sea, una licencia diferente a la anterior). Así, si es autor único de un programa podrá publicar una versión bajo una licencia de software libre y, si le desea, otra posterior bajo una licencia propietaria. En caso de existir más autores, y que la nueva versión contenga código cuya autoría les corresponda y que se vaya a publicar bajo otras condiciones, todos ellos deben dar el visto bueno al cambio de licencia.

Las licencias libres tienen tres modelos fundamentales. Las diferencias entre estos modelos están en cómo los propietarios de los derechos conceden parte de ellos a los usuarios bajo qué condiciones.

Dentro de las licencias más conocidas se encuentran la GPL, la LGPL, BSD, MIT, entre otras.

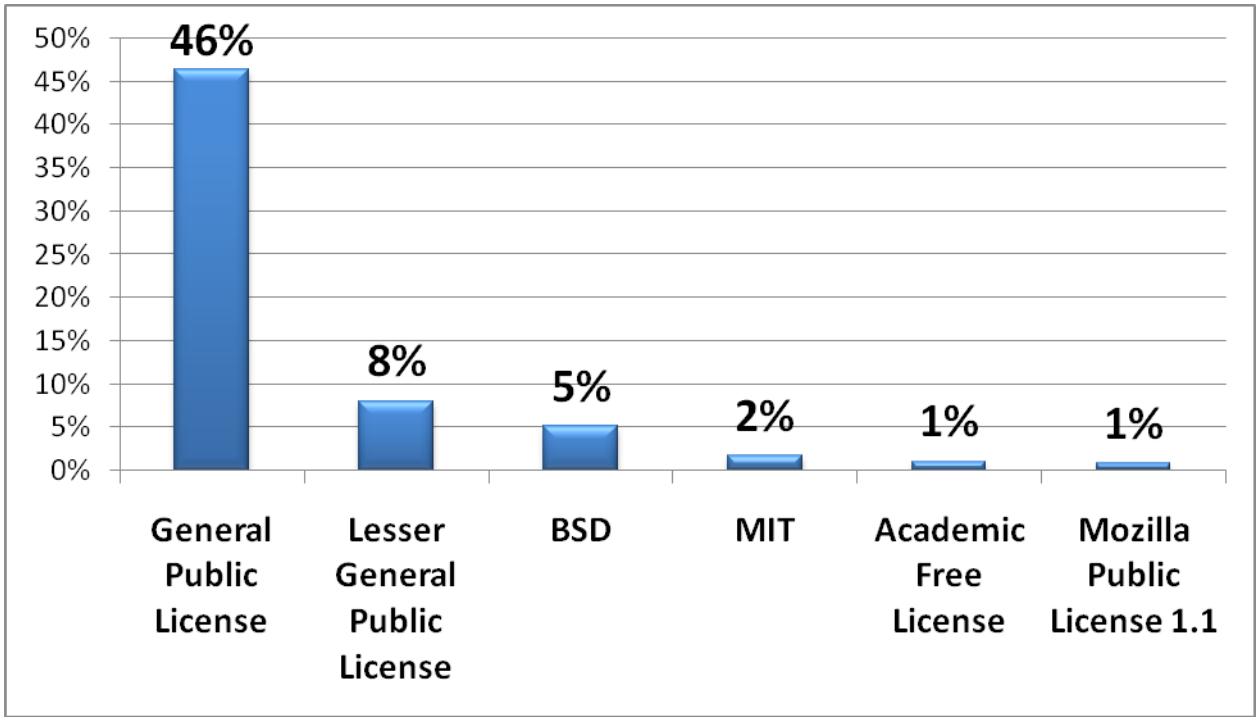

Figura 2: Utilización de licencias libres. [14]

- *Licencia Pública General de GNU (GNU-GPL)*

La licencia Pública General del proyecto GNU es la licencia más popular y conocida de todas las licencias del mundo del SWL. La GPL facilita la redistribución de ambos códigos, el código fuente y el ejecutable, aunque en el caso de que se redistribuya el binario, obliga a que también se permita y garantice el acceso al código fuente. Permite realizar modificaciones sin ningún tipo de restricciones, teniendo en cuenta siempre que solo se puede integrar código licenciado bajo GPL con otro código que se encuentre bajo una licencia similar o compatible. Esta característica hace que a esta licencia se le atribuya una consecuencia recursiva por garantizar la libertad del software al cual licencie, asegurando también que el código que sea publicado con este estado nunca pueda cambiar de condición o sea, nunca pueda ser propietario sin que no esto no implique una transgresión jurídica. Esta licencia se dice que maximiza las libertades de los usuarios.

- *Licencia Pública General Menor de GNU (GNU-GPL)*

Esta es otra licencia perteneciente a la Fundación del Software Libre (Free Software Foundation) pensada en sus inicios para el uso en bibliotecas, fue modificada recientemente para ser considerada la hermana menor de la GPL. La LGPL permite el uso de programas libres con software propietario. El programa en sí se redistribuye como si estuviera bajo la licencia GPL, pero con la permisiva de que se pueda integrar otro software sin limitaciones.

El uso de la LGPL solo se recomienda para casos específicos, por ejemplo cuando no hay otra alternativa y las librerías libres no resuelven el problema, entonces es cuando se recomienda su uso. Cuando la librería tiene funcionalidades extraordinarias respecto a las de su tipo y ofrece ventajas por sobre las demás, es preferible usar la GPL y no la LGPL. De esta forma se incentiva a los desarrolladores que quieran usar la librería a liberar también su código. [9]

- *BSD (Berkeley Software Distribution)*

Esta licencia tiene su origen en la publicación de versiones UNIX realizadas por la universidad de Berkeley en Estados Unidos. La única obligación que exige es la de da crédito a los autores, mientras que permite tanto la redistribución binaria, como la de los códigos fuentes.

En esta licencia se permite la redistribución en fuente y en binario con o sin modificación, siempre que se cumplan con las condiciones siguientes:

❖ Las distribuciones en fuente deben retener la nota de copyright y listar estas condiciones y la limitación de garantía.

❖ Las redistribuciones en binario deben reproducir la nota de copyright y listar estas condiciones y la limitación de garantía en la documentación.

❖ Ni el nombre del propietario ni de los que han contribuido pueden usarse sin permiso para promocionar productos derivados de este programa.

Esta licencia permite además que a partir de un programa distribuido bajo la licencia BSD pueda crear otro programa propietario, en fin. Se puede hacer (casi) lo que se quiera con el software. Por tanto, se maximizan las libertades de los desarrolladores. [9]

- *Licencia Pública de Mozilla (MPL)*

Esta licencia cumple plenamente la definición de SWL de código abierto de la Open Source Initiative (OSI), así como con las cuatro libertades enunciadas por la FSF. También deja abierto el paso a una posible reutilización de forma comercial no libre del SOFTWARE licenciado mediante ella, si quienes lo desarrollan lo deciden así; esto último pudiera ocurrir sin limitar la reutilización del código y sin tener que usar de nuevo la misma licencia.

Sin embargo la función principal de la MPL es ser utilizada como licencia de control para el navegador Mozilla (actual Firefox) y el SOFTWARE relacionado con este. También puede decirse que se usa frecuentemente por desarrolladores que quieren hacer libre su código fuente.

- *Apache*

Se puede decir que es una copia calcada de la licencia original, pero con modificación en todo lo referente a la autoría. Esta licencia incluye una cláusula adicional, como la imposibilidad de llamar las versiones redistribuidas de igual manera. Además de la prohibición de usar el nombre del propietario de los derechos para promocionar productos derivados.

De esta licencia existen tres tipos, siendo la 2.0 la más utilizada a la vez que considerada como una licencia de SWL. Las otras dos versiones carecen de copyleft, característica clave para no mantener la garantía de que resulte realmente SWL.

- *Creative Commons*

Fue fundada en el 2001 por expertos en propiedad intelectual, derecho en la sociedad de la información e informática, con el propósito de fomentar la existencia, conservación y accesibilidad de recursos intelectuales cedidos a la comunidad de diversas maneras. Su característica más sobresaliente es que permiten al autor seleccionar que tipo de libertades cede, además de la de copia, según cuatro dimensiones: dar crédito al autor original, permitir trabajos derivados, permitir redistribución comercial y permitir cambiar la licencia. Una de las licencias basada en Creative Commons es la de los cursos del MIT (MIT Open Courseware License Version 1.0) que obliga a dar crédito, impide el uso comercial y obliga además a conservar la licencia en trabajos derivados. [11]

Anteriormente se dijo que uno de los aspectos que debía revisarse de las licencias a usar en los proyectos es el de la compatibilidad. Si no se conocen bien las restricciones y las posibilidades de las licencias que se usen, puede que el proyecto no sea factible.

Existen algunas licencias consideradas libres que de cierta forma presentan incompatibilidad con la GPL. Universalmente se tiende a aceptar que esta última es la que más garantiza la definición GNU de libertad del software, lo que no necesariamente significa que sea la única ni la que mejor estimula el desarrollo y distribución del mismo. La licencia Apache presenta incompatibilidad con la GPL al exigir que se mencione explícitamente su nombre en los materiales de divulgación y promoción del nuevo producto que utiliza su código, así como en cada uno de los documentos titulares de derecho. Las cláusulas de la GPL no permiten cumplir esas exigencias de la Apache. Esto no implica que no se puedan usar simultáneamente o integrar programas de ambas licencias, pero en ese caso es imposible por ilegalidad la distribución del software derivado, al no poderse cumplir simultáneamente los requisitos de redistribución de ambas.

2.1.35. Propiedad Intelectual en proyectos de software libre: Copyleft

El copyleft es un instrumento legal similar al copyright, sin embargo sus objetivos son totalmente diferentes. Proviene de un juego de palabras en inglés, que indica la oposición de este término con los derechos de autor.

El copyright o derechos de autor se le aplican a creaciones particulares con el fin de restringir la copia y distribución no autorizada. En cambio, el copyleft no sólo limita la copia, modificación y distribución sino que promueve estas libertades, prohibiendo explícitamente cualquier acción orientada a restringirlas.

Si se realiza una obra bajo copyleft, tendrá que aplicárseles a las obras que se deriven de ella.

El término copyleft fue sugerido en primer lugar por Richard Stallman en 1984, y lo que pretendía evitar con él era que empresas con intereses comerciales se aprovecharan de software que se editaba bajo dominio público para hacer sus propias versiones y comercializarlas. [15]

El objetivo desde el principio no fue eliminar las leyes del copyright, sino crear una nueva licencia que reflejara los principios del copyleft.

2.1.36. *Acuerdo de licencia con los contribuyentes*

Es recomendable la elaboración de un acuerdo de contribuyentes a la licencia, el cual debe ser avalado por un abogado que definirá el grado de formalidad y seguridad para la firma.

Es habitualmente un compromiso electrónico que un desarrollador llena y reenvía a un proyecto, puede incluir firma digital o no. [15] Pero sí garantizando explícitamente al proyecto el derecho de usar el código de la contribución y que este realice acciones de acuerdo a la licencia que se defina. Estas acciones pueden ser reproducir, hacer trabajos derivados los cuales puede mostrar públicamente y poner bajo otras licencias para distribuirlos posteriormente además de las contribuciones.

Algo que se es importante para proyectos es que, cuando se solicite un acuerdo de contribución se especifique que no se está haciendo una transferencia de la propiedad intelectual y que éste no cambia los derechos a usar las contribuciones para cualquier otro propósito.

Los acuerdos deben reflejar los derechos que se le concederá al usuario final en la distribución de la licencia, además del derecho a la sublicencia. Se debe juntar la mayor cantidad de aspectos permisibles. Esto es con el fin de evitar la necesidad de volver a contactar al contribuyente que da la autorización para tomar alguna acción deseada con su contribución, como por ejemplo, la liberación en virtud de una licencia diferente.

2.1.37. Patentes

Se les llama a la autorización legal formulada por un gobierno o institución jurídica que permite al inventor excluir a otras personas de fabricar, utilizar o vender una creación, declarada como propia, durante el plazo de vigencia que se le otorgue. Este plazo está entre los 17 a 25 años. Las patentes promueven la investigación privada, sin valor monetario para el colaborador. EL individuo que posee una patente puede decide si permite el uso de la misma a otros y el precio que debe pagar por la licencia.

Por tanto se recomienda en cuanto a este tema que si alguna de las partes del programa que se esté realizando esté sustentada bajo alguna patente. Lo más sensato es que se abandone esa parte porque si ocurre alguna demanda por parte de los propietarios del programa patentado, no se podrá costear un proceso judicial porque los proyectos de código abierto no tienen el suficiente fondo monetario. Los defensores del sistema de patentes argumentan que este suscita o estimula la innovación, pero las fuertes restricciones que promueven en ocasiones lo que hacen es dificultarla de manera importante.

2.1.38. Aspectos financieros

El aspecto de la economía en el SWL es acerca de quién patrocina a los proyectos. En este epígrafe se exponen los principales modelos de negocio que se ponen en práctica en cuanto al financiamiento de los proyectos.

A continuación se presentan varias formas de financiamiento externo en el que no todo el trabajo realizado es voluntario. No quiere decir que esa sea la única forma de conseguir recursos que tienen los proyectos de código abierto. Hay otras formas, está el caso de los desarrollos por voluntarios solamente, que será discutido más adelante.

En los casos donde hay un algún flujo de capital externo que se encargará de dar los recursos para su desarrollo. Se le considera al software construido como parte de esa financiación.

Financiación pública

En este caso la entidad financiadora puede ser un gobierno (local, regional) o una institución pública. En la mayoría de los casos no se encuentra la financiación explícitamente en productos o servicios relacionados con el SWL, sino que constituyen parte de algún contrato con fines más generales. Las motivaciones para este tipo de financiación son muy variadas, entre ellas se encuentran:

- Científica

Su objetivo principal no es el de producir software, sino el de más bien, investigar en algún campo, no necesariamente relacionado con la informática. Es muy posible que para esto sea necesario desarrollar herramientas que ayuden a alcanzar las metas del proyecto.

- Social

El SWL se considera una herramienta de gran interés para la creación de la infraestructura básica para la sociedad de la información. Las entidades interesadas en financiar proyectos para ayudar al acceso universal pueden financiar proyectos relacionados con el SWL.

Financiación privada sin ánimos de lucro

Es un tipo de financiación muy parecido al anterior, lo que esta es realizada por instituciones no gubernamentales. En estos casos, la motivación principal puede ser el de producir SWL para un ámbito en el que la entidad financiadora considere relevante.

Financiación por quien necesita mejoras

Este es el caso donde la entidad financiadora necesita mejoras en un producto libre. De acuerdo a la situación que se le presente, la entidad financiadora solicita el servicio que necesita ya sea en la corrección de errores o la necesidad de uso de una cierta funcionalidad.

Financiación con beneficios relacionados

En este caso lo que busca la entidad financiadora es conseguir beneficios en productos relacionados con el programa a cuyo desarrollo aporta recursos.

Algunos de los productos relacionados con el software son libros, el hardware de dicho programa, cd con programas, entre otros.

Financiación como inversión interna

Es cuando el software se desarrolla para satisfacer las necesidades internas de la empresa. Y las ventajas que aporta son la de los beneficios relacionados cuando el software se desarrolla para satisfacer las necesidades de la empresa o se decida liberar el código para abrir una línea de producción a partir del mismo. [9]

Conclusiones

En este capítulo se desarrollaron los puntos que conforman la estrategia que se pretende sea aplicada a los proyectos de código abierto próximo a realizar su lanzamientos. Se detalló cada aspecto con sus variantes, ventajas y desventajas de forma tal que sirvan como referencia inicial a todos los aspectos que deben ser tenidos en cuenta para el lanzamiento de proyectos de código abierto.

Capítulo III Valoración de la Propuesta

Al exponer los métodos utilizados en esta investigación se reseñó de manera general el método empírico diseñado y aplicado para diagnosticar el estado o nivel de aplicación que tienen los aspectos referidos en la encuesta en los proyectos de la Facultad 10 de la UCI. Se mencionó además el método matemático y estadístico para el procesamiento e interpretación de los datos empíricos obtenidos.

Para la valoración y aceptación de la propuesta planteada en el objetivo general, se utilizó como criterio las opiniones de un grupo de especialistas. Este panel se conformó con varios profesionales, que poseen conocimientos sobre los proyectos en cuestión y se desempeñan como administradores de proyectos comunitarios.

En el presente capítulo se hará la descripción de los pasos utilizados en la selección del grupo de especialistas que harán la evaluación del contenido. Para el proceso de evaluación se tomó en cuenta el proceso de selección de especialistas, posteriormente la elaboración de la encuesta y por último los resultados de la evaluación.

3.1.1. Proceso de selección de especialistas

Se entiende por experto a una persona que es capaz de emitir opiniones acerca de un campo determinado porque posee conocimientos y experiencias suficientes de dicho campo. Además de interpretar correctamente las informaciones del tema referido además de ofrecer valoraciones y hacer recomendaciones al respecto.

En el desarrollo de este proceso se consideraron varios aspectos esenciales:

- Determinar la cantidad de especialistas a encuestar (población).
- Conformar el listado de los especialistas.
- Confirmar la participación de los especialistas.
- Procesamiento cualitativo y cuantitativo de las respuestas.

Se tiene en cuenta que las entidades de los especialistas no son conocidas y las respuestas emitidas por cada uno de ellos tampoco.

3.1.1. Determinar la cantidad de especialistas a encuestar

Para este punto se debe configurar el panel, que se plantea sea de diez a veinte especialistas que se desempeñan como administradores de proyectos de código abierto, no se tomó mucho en cuenta

la cantidad de especialistas, y sí la diversificación de las respuestas. Partiendo de esto se escogió un panel de quince especialistas teniendo en cuenta el nivel de las preguntas a responder.

3.1.2. Conformar el listado de los especialistas

La selección de los especialistas propicia una mayor posibilidad de obtener resultados claros y precisos. Da una visión de cómo los proyectos se preparan para la salida a la comunidad.

La confección del listado se realizó atendiendo a la posibilidad de participación de estos administradores, además poseen conocimientos de temas que son propios de los proyectos como son el manejo del personal involucrado, la infraestructura tecnológica y algunos de los aspectos que se consultan al iniciar cualquier nuevo proyecto.

Las cualidades que caracterizan a los seleccionados son:

- Responsabilidad.

A la hora de medir el tiempo de entrega de la encuesta.

- Seriedad.

En la consistencia de las respuestas emitidas.

- Capacidad de análisis.

La capacidad de interpretación de acuerdo a los temas relacionados en la encuesta.

- Honestidad.

Cualidad importante debido a la seriedad de esta encuesta, para conocer el verdadero estado que tienen los proyectos productivos en la universidad.

Estas cualidades han permitido que las opiniones obtenidas cuenten con un cierto grado de credibilidad y sean válidas para el objetivo propuesto con esta encuesta.

3.1.3. Confirmar la participación de los especialistas

Una vez conformado el listado, se reunió con cada especialista y se le explicó en qué consistía la encuesta y el objetivo que se perseguía con la misma. Cuando es recibida la respuesta de aceptación se conforma el listado oficial de los especialistas que participarían en la investigación. De esta forma culmina el proceso de evaluación.

3.1.4. Procesamiento estadístico y cualitativo de las respuestas

El procesamiento estadístico de las encuestas se realizó mediante el tanto por ciento. Y el cualitativo es mediante las opiniones emitidas por los encuestados. Los resultados obtenidos se relacionan en el epígrafe 3.3, permitiendo perfeccionar o remodelar la propuesta antes de incluirla en la práctica.

3.1.2. Elaboración de la encuesta

La encuesta se lleva a cabo de una forma anónima, con el objetivo de medir el grado de factibilidad o impacto de la propuesta, sus ventajas y desventajas, además de los posibles inconvenientes que se pueden presentar en su introducción en la práctica de los proyectos de código abierto. Obteniendo en qué medida se tienen en cuenta los aspectos expuestos para la elaboración y lanzamiento de proyectos de código abierto a la comunidad.

Para su elaboración se tuvieron en cuenta los aspectos que se debían cumplir por parte de los proyectos de código abierto para su salida a la comunidad.

Contiene catorce preguntas referentes a la factibilidad de introducir la propuesta en la práctica de los proyectos comunitarios de código abierto a partir de indicadores predeterminados. Las preguntas se dividen en bloques dando un orden lógico a los pasos que se deben seguir a la hora de hacer el lanzamiento.

En el primer bloque se encuentran las preguntas que se relacionan con los puntos que se deben revisar al iniciarse el proyecto. Cuenta con un total de cinco. En el segundo bloque entran las preguntas que se refieren a cuando el proyecto ya está en marcha. Con un total de cuatro preguntas con seis incisos. El tercer bloque contiene las preguntas complementarias que son las que se refieren a los aspectos que le dan al proyecto las últimas pautas a seguir para consolidarse como tal. Cuenta con tres preguntas y tres incisos. El cuarto bloque está compuesto por una pregunta y es la que se refiere a los medios utilizados para hacer el lanzamiento del proyecto. El último bloque es conformado por una sola pregunta y se refiere a la opinión de incluir la propuesta diseñada a la guía estratégica para el lanzamiento de proyectos de código abierto y que estos sean exitosos.

Se reunió con cada especialista y se le explicó la encuesta conjuntamente con el objetivo que perseguía la misma para que se entendiera la importancia de que se entregara en tiempo.

Ver encuesta en el anexo uno.

3.1.3. Resultados de la evaluación

A la hora de evaluar los resultados de la encuesta se conformaron una serie de parámetros anteriormente mencionados, los parámetros se enuncian a continuación:

- Aplicación actual en los proyectos de la comunidad universitaria.

Da la medida de cómo están aplicados los aspectos referidos en la encuesta por parte de los proyectos de la universidad.

- Adaptabilidad a los proyectos de código abierto.

De acuerdo a las opiniones emitidas por los encuestados, se obtiene una visión de si los puntos de la estrategia pueden ser adaptados a los proyectos de acuerdo a sus características de infraestructura.

- Posibilidad de aplicación en los proyectos de la UCI.

Es de acuerdo a las consideraciones de los especialistas si esta propuesta tiene posibilidad de ser implementada en la universidad.

- Necesidad de aplicación en los futuros proyectos comunitarios.

Este punto es lo que afirma la importancia de la estrategia propuesta por todos los aportes que implica a la hora de lanzar un proyecto a la comunidad.

Ver opiniones emitidas en el anexo dos.

3.1.4. Aplicación actual en los proyectos de la comunidad universitaria

Se realizó un análisis para determinar cómo los especialistas ponen en práctica los aspectos relacionados en la encuesta, los resultados del bloque que representa los aspectos iniciales.

Primer bloque.

Bloque que relaciona los aspectos iniciales, las filas Especialistas relacionan los integrantes de los proyectos que intervienen en la encuesta; la fila Uso (%) es la que indica cómo los aspectos relacionados son implementados.

Especialistas	**1**	**2**	**3**	**4**
Uso (%)	75	0	100	75
Especialistas	**5**	**6**	**7**	**8**
Uso (%)	+100	75	50	100

Especialistas	**9**	**10**	**11**	**12**
Uso (%)	75	75	+100	100
Especialistas	**13**	**14**	**15**	**16**
Uso (%)	+100	50	100	75
Especialistas	**17**	**18**		
Uso (%)	100	50		

Tabla 1: Comportamiento del primer bloque de la encuesta en la muestra de los proyectos.

La tabla anteriormente expuesta da una idea de cómo son implementados algunos de los aspectos iniciales de acuerdo a las opiniones de los especialistas. Se observa además que un 27.8 % de los proyectos incluyen en su estrategia aspectos como: definición de la arquitectura, procesos de formación de los integrantes del proyecto, disponibilidad de desarrolladores, revisión del entorno local, incluyen además la revisión de las necesidades personales de los desarrolladores. La implementación en general de los aspectos en la muestra escogida es de un 76.4 %.

Segundo bloque.

Bloque que relaciona los aspectos de infraestructura relacionados en la encuesta, como se explicó anteriormente, las filas Especialistas relacionan los administradores de los proyectos y las de Uso (%) son las que señalan la medida en que son implementados por parte de los proyectos. Cabe destacar que este segundo bloque es el que presenta más problemas en cuanto a su implementación porque no son implementados todos los aspectos siendo este bloque el más importante porque relaciona los aspectos de mayor peso en la encuesta.

Especialistas	**1**	**2**	**3**	**4**
Uso (%)	62.5	87.5	75	50
Especialistas	**5**	**6**	**7**	**8**
Uso (%)	58.3	45.8	45.8	54.2

Especialistas	**9**	**10**	**11**	**12**
Uso (%)	45.8	25	62.5	70.8
Especialistas	**13**	**14**	**15**	**16**
Uso (%)	62.5	37.5	45.8	79.2
Especialistas	**17**	**18**		
Uso (%)	58.3	54.2		

Tabla 2: Comportamiento del segundo bloque de la encuesta en la muestra de los proyectos.

Los resultados que se reflejan en la tabla anterior muestran que la mayoría de los proyectos no cumplen con aspectos (empaquetamiento, documentación, voluntarios, infraestructura tecnológica) que sirven para el mejor funcionamiento una vez puestos en marcha. Todo esto es avalado por los resultados obtenidos anteriormente que indican que el segundo bloque de la encuesta es desarrollado sólo un 56.7 %, índice que muestra el bajo desarrollo de los proyectos por lo que en estas condiciones están muy lejos de ser proyectos exitosos.

El bajo por ciento en que son tenidos desarrollados aspectos de la encuesta como son:

Sitios web: la mayoría de los proyectos manifiestan no tener un sitio web porque no es posibilitado por la infraestructura interna de la universidad.

Manejo de voluntarios: No en todos los proyectos se hace la delegación de tareas, en muchos si hacen las convocatorias para añadir nuevos integrantes al proyecto y estos los ubican de acuerdo a los conocimientos que tengan los nuevos usuarios.

Documentación: En la mayoría de los proyectos encuestados no se relacionan este aspecto los tipos de documentos que realizan en el proyecto. Hay una escasez de documentación, es muy poca, sólo se refieren a la que ponen en el GForge para interés de los integrantes del proyecto.

Tercer bloque.

Especialistas	**1**	**2**	**3**	**4**
Licencia	GPL	GPL	GPL, LGPL	GPL, LGPL

Especialistas	**5**	**6**	**7**	**8**
Licencia	GPL	GPL, LGPL	GPL, LGPL	GPL
Especialistas	**9**	**10**	**11**	**12**
Licencia	BSD, GPL	LGPL	GPL, CC	GPL
Especialistas	**13**	**14**	**15**	**16**
Licencia	GPL	GPL	GPL, LGPL	GPL, LGPL, BSD
Especialistas	**17**	**18**		
Licencia	GPL	GPL		

Tabla 3: Uso de las licencias en los proyectos encuestados.

Para el análisis del tercer bloque también se tuvieron en cuenta las entrevistas que se le realizaron a algunos de los encuestados con el fin de de emitir un diagnóstico acerca del uso de las licencias y del conocimiento que se tienen de los aspectos legales en los proyectos.

En el desarrollo del capítulo anterior se exponen varias licencias utilizadas en proyectos de software libre, pero algunos especialistas refieren a que ellos trabajan bajo la licencia que la universidad designe sea mejor para los proyectos que en ella se desarrollan.

Otra conclusión derivada de las entrevistas es que en la mayoría de los proyectos hay conocimientos sobre los aspectos legales del SWL (licencias, propiedad industrial, copyleft).

3.1.5. Adaptabilidad a los proyectos de código abierto

Se decidió analizar este parámetro porque determina en qué medida los proyectos son capaces de implementar los aspectos de la estrategia. Como resultado del epígrafe anterior se comprobó la situación actual en la que se encuentran los proyectos con relación a los aspectos propuestos, además de conocer las cosas buenas que los caracterizan así como las que quedan por profundizar. En las entrevistas realizadas a los encuestados se pudo observar que los administradores de los mismos conocen de la existencia de los aspectos propuestos y simplemente no los desarrollan porque no tienen nada que los obligue hacerlo provocando que muchas veces sean olvidados, pues

se le da más importancia al código que a otros elementos que aunque parezcan innecesarios son imprescindibles para que el proyecto funcione correctamente y tenga una mejor organización en su salida a la comunidad. Por tanto se puede concluir que los aspectos planteados pueden ser implementados por los proyectos, siguiendo paso por paso la estrategia propuesta evitando que se cometan estos errores y logrando el éxito una vez realizado el lanzamiento a la comunidad.

3.1.6. Posibilidad de aplicación en los proyectos de la UCI

Este punto se analizó de acuerdo a las opiniones obtenidas en las entrevistas realizadas a algunos encuestados. El tema de la universidad se considera algo complicado por parte de los proyectos cuando se observa de una forma individual. Primero se debe asegurar la base tecnológica para que se puedan desarrollar algunos de los aspectos referidos en la estrategia. Se observó la limitación en cuanto a la creación sitios web de cada proyecto externos a las plataformas de desarrollo.

También se debe fomentar más la cultura de trabajo colaborativo en todos los estudiantes, además de hacer énfasis en la necesidad que hay de lanzar proyectos exitosos a la comunidad.

3.1.7. Necesidad de aplicación en los futuros proyectos comunitarios

El último bloque de la encuesta se explicó anteriormente que serviría para determinar si la propuesta desarrollada era necesaria. Las opiniones emitidas lo demuestran, las mismas se refieren a la inclusión de este estudio en los lanzamientos de los proyectos por los diversos aportes que el mismo brinda. El ciento por ciento de los especialistas encuestados ven la aplicación de la estrategia de una forma positiva.

Lo obtenido refleja que la estrategia planteada cumple con muchos aspectos que son de desconocimiento en los proyectos y por tanto no son aplicados, así que los proyectos están trabajando sin tener en cuenta muchos puntos que le sirven de base para llegar a consolidarse y ganar un lugar entre la comunidad universitaria.

Si lo que se desea es lanzar proyectos lo suficientemente robustos como para servir de apoyo a la estrategia de migración del país, entonces se recomienda el estudio exhaustivo de las posibilidades que brinda la guía estratégica anteriormente propuesta, además de tomarla como un apoyo a la hora de adaptar y/o implementar en el proyecto.

Conclusiones

Los resultados obtenidos con la encuesta anteriormente explicada, demostraron la necesidad de poner en práctica este estudio en los proyectos de código abierto próximos a realizar su lanzamiento.

Capítulo III Valoración de la Propuesta

Después de haber realizado la evaluación siguiéndose por los parámetros descritos, se puede llegar a la conclusión que se ha conformado una propuesta que servirá de base a cualquier proyecto libre para organizarse en aspectos recogidos en la guía.

Se espera que con esta investigación se mejoren los resultados a la hora de sacar un proyecto nuevo a la comunidad universitaria y por consiguiente ayude a la migración del país a otro sistema operativo con proyectos de alta calidad.

Conclusiones Generales

Como solución a la problemática que se investiga fue posible diseñar una guía estratégica para apoyar y potenciar el lanzamiento de proyectos de código abiertos, examinando como hacer bien las cosas desde el inicio de cualquier proyecto hasta la consolidación de sus objetivos.

La revisión bibliográfica elaborada permitió exponer la teoría sobre el tema que se investiga y poderlos expresar en diferentes epígrafes.

Fue posible ofrecer una explicación detallada de cada uno de los aspectos que conforman la guía estratégica propuesta para un mejor entendimiento de cada aspecto y la importancia que tiene cada uno de ellos para el proyecto.

El ciento por ciento de los especialistas consultados considera de gran utilidad la guía estratégica elaborada por los aportes al conocimiento de administradores y miembros de proyectos comunitarios de código abierto para una mejor consolidación de sus proyectos.

Recomendaciones

Para lograr el buen desempeño en la salida de los proyectos de código abierto a la comunidad se recomienda:

- La revisión de esta propuesta por parte de los proyectos comunitarios desarrollados actualmente en la facultad 10.
- Tener en cuenta este estudio para dar apoyo a la salida de proyectos comunitarios en otras áreas de la universidad.
- A través de la difusión de esta propuesta contribuir a aumentar la cultura comunitaria en la universidad.
- Mejorar la propuesta diseñada en las plantillas que fueron elaboradas para ser llenadas por los nuevos proyectos.
- Enriquecer la guía estratégica elaborada y trabajar en su inclusión dentro de las plataformas de desarrollo colaborativo.

Bibliografía

REFERENCIADAS

[1] 1. **Batista**, **Gilberto García**. *Fundamentos de las Ciencias de la Educación. Maestrías en Ciencias de la Educación*. La Habana: Pueblo y Educación. Vol. II. ISBN 959-13-1428-0.

[2] 2. **Rabasa**, **Laya del C.**. *Aspectos claves para el lanzamiento de proyectos de código abierto*. Memorias de la Convención Informática 2009. Ciudad Habana. 2009.

[3] 3. **Blasco**, **Miguel Guinalíu**. 5campus.org. [Online] 2003. [Cited: Febrero 24, 2009.] <http://www.ciberconta.unizar.es/leccion/comunidades>.

[4] 4. **Valiente**, **Francisco Javier**. *Comunidades Virtuales en el Ciberespacio*. Universidad San Pablo CEU de Madrid.

[5] 5. **Salinas**, **Jesús**. *Comunidades Virtuales y Aprendizaje Digital*. España: Universidad de las Islas Baleares.

[6] 6. **Martín**, **Manuel Alejandro Gil**. *Proceso de desarrollo de aplicaciones de acuerdo a los conceptos de software libre. Aplicación en la Universidad de las Ciencias Informáticas*. La Habana: s.n., 2007.

[7] 7. **Altamiranda**, **M.sc**. **Junior**. *GFORGE*. Mérida: s.n., 2007. 2005000170.

[8] 8. **Martínez Sánchez**, **Yanexis**, **Correoso Barroso**, **Daymeirelis**. *Desarrollo Colaborativo con la Herramienta GForge*. Universidad de las Ciencias Informáticas. La Habana, 2007. 75

[9] 9. **Robles**, **Gregorio**, **González Barahona**, **Jesús**, **Seoane Pascual**, **Joaquín**. *Introducción al Software Libre*. Barcelona: s.n., Noviembre, 2003. B-38-682-2003.

[10] 10. *Libre, Comunidad de Software*. Portal de Software Libre de la UCI. [Online] http://softwarelibre.uci.cu.

[11] 11. Fogel, Karl. Producing Open Source Project: How to Run a Sucessful Free Software Project. 2005.

[12] 12. www.selenic.com. [En línea] http://www.selenic.om/mercurial.

[13] 13. git.or.cz. [En línea] http://git.or.cz.

[14] 14. **Soler**, **Otto Batista**. *La propiedad intelectual y la industria del software*. [Presentación Digital] La Habana: Universidad de las Ciencias Informáticas, 2008.

[15] 15. **Morfeo**, Comunidad. Sitio Oficial de la Comunidad Morfeo. [En línea] [Citado el: 29 de Abril de 2009.] http://forge.morfeo- project.org/wiki/index.php/Gesti %C3%B3n_de_contribuciones_en_la_Comunidad_MORFEO.

CONSULTADAS

Aguilar, **Luis**. linux.sys-con.com. [Online] 2005. [Cited: enero 17, 2009.] http://linux.sys-con.com/read/48557.htm.

Sopena, Diccionario Enciclopédico Ilustrado, 1965, pág. 874, pág. 3697.

Fogel, **Karel**. *Producing Open Source Project: How to Run a Sucessful Free Software Project*. 2005.

Nuñez Rizo, **Anaisa**, **Peña Cruz**, **Susel**. *Propuesta de una Línea de Producción para el proyecto Nova Linux*. Universidad de las Ciencias Informáticas. La Habana, 2008. 117

Feller,**Joseph**, **Fitzgerald**, **Brian**, **Hissam**, **Scott A**., **Lakhani**, **Karim R**. *Perspectives on Free and Open Source Software*. Cambridge, Massachusetts. 571

Corzo, **Elizabeth López.** *UCI: Camino hacia el software libre*, 2008 [Disponible en: http://www.cubasi.cu/desktopdefault.aspx?spk=160&clk=227366&lk=1&ck=116352&spka=35]

Merconchini ,**David Grau**. *Informática y nuevas tecnologías. Software libre II: Una estrategia decisiva de desarrollo*, 2008. [Disponible en: http://www.juventudrebelde.cu/cuba/2008-02-14/software-libre-ii-una-estrategia-decisiva-de-desarrollo/]

Software Libre. [Online] [Cited: Febrero 4, 2009.] http://www.softwarelibrecr.org/faq/comunidad.

Wikipedia. [Online] [Cited: Diciembre 3, 2008.] http://es.wikipedia.og/wiki/Trabajo_colaborativo.

Soro, **Emilio Saez**. *Ensayo de una metodología de estudio de las comunidades virtuales.* s.l. : Universidad Jaumel I, 2006.

Subversion. [En línea] [Citado el: 20 de 3 de 2009.] http://subversion.tigris.org/.

Ubuntu. [En línea] http://www.ubuntu.org.

bazaar-ng.org. [En línea] Marzo de 2009. http://bazaar-ng.org.

venge.net. [En línea] http://www.venge.net/monotone/.

codeville.org. [En línea] http://codeville.org/.

gnu.org. [En línea] Marzo de 2009. http://www.gnu.org/software/gnu-arch/.

moodle.org [En línea] [Citado el:29 de Abril de 2009] http://docs.moodle.org/es/Documentaci%C3%B3n_para_Desarrolladores

Gé Pérez, **Surima**, **Causse Ascanio**, **Yumislaidi**. *Propuesta de implementación de CMMI en el área de proceso: Aseguramiento de la Calidad de los Procesos y Productos, para los proyectos productivos de la Facultad 2*. Universidad de las Ciencias Informáticas. La Habana, 2008. 97

Anexos

Anexo I "Encuesta realizada a los administradores de los proyectos"

Encuesta para determinar en qué medida se tuvieron en cuenta los siguientes aspectos para la elaboración y lanzamiento a la comunidad de proyectos de código abierto.

(Marque con una X su respuesta)

1- Al iniciarse un proyecto de desarrollo para la comunidad ¿cuál o cuáles de los siguientes aspectos usted considera que se tuvieron en cuenta:

1. ____Revisión del Panorama Internacional
2. ____Misión del Proyecto
3. ____Soporte tecnológico para Hospedaje del Proyecto
4. ____Disponibilidad del Proyecto (facilitar las descarga e instalación)
5. ____En caso de otros .Enúncielos.

_____

_____

2- Una vez que ya se han dado las primeras ideas del proyecto y puesto en marcha: ¿Cuál o cuáles de los siguientes se tuvieron en cuenta?

2.1 Discusión de cómo hacer el empaquetamiento del proyecto

Actualizaciones ☐ Versiones ☐

2.2 Proyección de la Documentación para

Usuarios ☐ Desarrolladores ☐

Instalación ☐ Comunidad (wikis) ☐

2.3 Infraestructura tecnológica

- ☐ Presentación del proyecto
  - Sitio Web ☐
  - Noticias ☐
  - Temas en foro ☐
  - Otros ☐ ¿Cuál o Cuáles?:_____
- ☐ Canales de comunicación
  - Chat ☐
  - Listas de correo ☐
  - Foros ☐
  - Otros ☐ ¿Cuál o Cuáles?:_____
- ☐ Controlador de versiones
  - Especifique ¿Cuál o Cuáles?:_____
- ☐ Gestión de fallos
  - Especifique ¿Cuál o Cuáles?:_____
- ☐ Seguimiento de tareas
  - Especifique ¿Cuál o Cuáles?:_____

2.4 Manejo de voluntarios

¿Cómo involucrarse en la comunidad?

_____
_____

¿Cómo contribuir con la comunidad?

_____

_____

Se realiza la asignación de roles en la comunidad Si ☐ No ☐

Si usted asigna roles en la comunidad. Describa cómo asigna los roles de los involucrados en el desarrollo del proyecto

_____
_____

Principales roles:
- Desarrolladores ☐
- Programadores ☐
- Traductores ☐
- Distribuidor de tareas ☐
- Documentador técnico ☐
- Evaluadores de documentación ☐
- Otros ☐ Cuál o Cuáles?:_____

3- Valoración de otros aspectos.

Se tiene conocimiento de los aspectos legales de software libre Si ☐ No ☐

Se realizan consultas legales con algún abogado Si ☐ No ☐

3.1 Aspectos legales

Licencias:
- ☐ LGPL
- GPL ☐
- BSD ☐
- MIT ☐
- No definida ☐
- Otros ☐ Cuál o Cuáles?:_____

3.2 Aspectos financieros

- Patrocinado por: _____

- Fuentes de financiamiento: _____

3.3 Estado de desarrollo

Publicaciones de avances periódicos: Sí ☐ No ☐

4- Medios utilizados para el lanzamiento del proyecto.

_____
_____
_____
_____

5- Cuál es su opinión sobre el hecho de incluir este estudio en el lanzamiento de proyectos de código abierto a la comunidad.

_____
_____
_____
_____

Anexo II "Opiniones emitidas por los especialistas"

Especialista 1

Sirve de apoyo para la visión o concepción del Proyecto.

Especialista 2

Considero que todos estos aspectos son de vital importancia a la hora de lanzar un software a la comunidad. Se deben continuar implementando.

Especialista 3

Podrá ser un buen punto de partida para futuros lanzamientos, por lo que me parece una buena idea.

Especialista 4

Puede convertirse en una guía para los desarrolladores que deseen crear un nuevo proyecto.

Especialista 5

Es importante para la generalidad de la comunidad, las características de los proyectos y principales necesidades e intereses de los mismos.

Especialista 6

Estoy en total acuerdo con lo planteado, de hecho hay proyectos hoy que están trabajando y no tienen algunos de los conocimientos que se piden en la estrategia y que deben incluir.

Especialista 7

Puede dar un acercamiento sobre los aspectos que más importancia tengan para el proyecto, así como asegurar que condiciones necesarias e indispensables sean otorgadas.

Especialista 8

En mi opinión los proyectos de código abierto necesitan de una ayuda para estructurarse ya que en la mayoría de los casos están compuestos solamente por desarrolladores y omiten documentación y demás aspectos.

Especialista 9

Sería muy bueno incluir y aplicar esta propuesta en los proyectos ya que es un estudio para hacer lanzamientos de nuevos proyectos de código abierto a la comunidad, con el cual se obtendría mucha información a la hora de crear un proyecto, motivaría a los usuarios a compartir sus ideas dentro de la comunidad, aportándole mucha más información y proporcionándole nuevas oportunidades a los involucrados o los posibles voluntarios.

Especialista 10

Yo lo veo de manera positiva, como una base, medio de apoyo para los líderes de proyectos productivos de código abierto. Muy útil si se desea integrar a la comunidad en la vida de un proyecto productivo.

Especialista 11

Este estudio puede impulsar favorablemente la concepción y desarrollo de los proyectos de software libre y código abierto en la UCI contribuyendo a la mejor organización de los mismos.

Especialista 12

Este estudio da la posibilidad de conocer la actividad productiva basada en comunidades. Pudiera servir de base para implementar o estandarizar estrategias de desarrollo de este tipo.

Especialista 13

Este estudio da la posibilidad de conocer la actividad productiva basada en comunidades de desarrollo. Si se hace con gran cantidad de proyectos pudiera servir para implementar o estandarizara una estrategia para este tipo de desarrollo.

Especialista 14

Pudiera servir de enlace a futuras propuestas de proyectos de este tipo. Orientara s además a la institución en materia de nuevas ideas de desarrollo.

Especialista 15

Pienso que pudiera mejorar la manera en que contribuimos entre todo, así como darle prioridad en mejorar los servicios que más se utilizan en el sitio donde se hospedan los proyectos.

Especialista 16

El estudio debe incluirse o tener en cuenta cuando se desee lanzar un proyecto comunitario. Se debe lograr con este estudio una estrategia para la creación de proyectos de SWL.

Especialista 17

El trabajo me parece bueno pero se enfrenta a las políticas de licenciamiento de los productos de la universidad de las ciencias informáticas que son muy restrictivas.

Anexo III "Aspectos iniciales"

<Nombre del Proyecto>

Aspectos iniciales

Versión <#>

Historia de revisiones

Fecha	Versión	Descripción	Autor
<dd/mm/aaaa>	<#>		Responsable (s)

1. Introducción

[Se da una breve introducción de lo que significa la planilla para los proyectos que pretendan hacer su lanzamiento. Debe indicarse la importancia que se le concede a los aspectos que van a dar la primera impresión a los usuarios con tal de adquirir integrantes potenciales.

Se expresa cómo el proyecto va a poner en práctica estos aspectos.]

2. Panorama Internacional

[Para este aspecto se deben revisar si existe algún proyecto que esté haciendo lo mismo que se propone hacer con el suyo. Si existe alguien haciéndolo y es bajo una licencia libre, no es necesario empezar desde cero. En este caso se debe especificar el nombre del proyecto y los objetivos del mismo, si se pretende integrar a este u obtener alguna ventaja del mismo.

En caso de que el objetivo del proyecto que se quiera abrir sea educativo y no requiera del código pre-existente, entonces se debe especificar la existencia de los proyectos que coincidan, pero esclarecer que no se va a tomar nada de estos proyectos.]

3. Preparar al proyecto para la versión pública

[Se deben llenar con los pasos que realiza el proyecto para cambiar de versión. Cómo se definieron los objetivos y los posibles desacuerdos que surgieron en este tiempo. El empaquetamiento para el público y los ficheros dedicados a los nuevos integrantes con tal de que comprendan el mismo.]

4. Nombre

[El nombre no es lo que define un proyecto pero tampoco hay que descuidar este punto, ya que da una pequeña visión de lo que significa el proyecto.]

5. Misión

[Después del nombre esta es la segunda característica que los nuevos usuarios buscan para entender el proyecto. Esta debe estar en una parte prioritaria de la página donde se va a anunciar el proyecto. Debe ser corta, concreta y sencilla.]

6. Carácter del proyecto

[Esta declaración debe ponerse en la página principal del proyecto, además de declarar que el proyecto es open source, debe especificarse bajo qué tipo de licencia se distribuyen las aplicaciones.]

7. Lista de características y requerimientos

[Se deben listar los requerimientos que necesita el proyecto. En el orden de planeados, en progreso y cumplidos.]

8. Anuncio del proyecto

[Si se ha hecho el anuncio del proyecto mediante un sitio web, especificar la dirección y el nombre del portal. Si es mediante otro medio de difusión, (correo, intranet) especificar cuál.]

9. Hospedaje

[Declarar donde va a estar alojado el proyecto, si es en sitio externo decir a que organización pertenece y poner la dirección. En caso de que sea el sitio propio del proyecto, poner la dirección.]

10. Disponibilidad

[Se debe especificar cómo se puede obtener el proyecto (código fuente, ejecutable), y la dirección donde se pueda descargar.]

11. Presentación

[Especificar de qué forma el proyecto va a ser su presentación ante los usuarios. Si es mediante un sitio web y las características del mismo. Si es en forma de noticias, los canales por los cuales se difunden las mismas, etc.]

12. Estado de desarrollo

[Publicar en qué estado se encuentra el proyecto y las acciones que hacen para darlo a conocer.]

Anexo IV "Aspectos de documentación"

<Nombre del Proyecto>

Aspectos de documentación

Versión <#>

Historia de revisiones

Fecha	Versión	Descripción	Autor
<dd/mm/aaaa>	<#>		Responsable (s)

1. Introducción

[Se da una breve introducción de lo que significa la planilla para los proyectos que pretendan hacer su lanzamiento. Debe indicarse la importancia que se le concede a los aspectos que tienen que ver con la documentación del proyecto. Se expresa cómo el proyecto va a poner en práctica estos aspectos.]

2. Documentación

[Especificar qué tipo de documentación va a ser desarrollada en el proyecto. Además de mencionar los manuales que se van a poner a disposición y para qué tipo de usuarios (usuarios comunes, desarrolladores) están hechos.]

2.1. Documentación para usuarios

[Especificar qué documentación se tienen para los usuarios (manuales de cómo interactuar con la aplicación o con el proyecto.)]

2.2. Guías para desarrolladores

[Especificar cuáles son las guías que son específicamente para los desarrolladores, son las que promueven las reglas para trabajar en el proyecto (guías de estilo, cómo trabajar con la plataforma, etc.)]

2.3. Documentación para desarrolladores

[En este apartado van los manuales que le dan a los desarrolladores cómo desenvolverse en el proyecto (cómo trabajar con el código, los comandos que se utilizan y su aplicación, entre otros.)]

3. Empaquetamiento, actualizaciones y versionado

[Se debe especificar cómo va a ser el empaquetamiento del proyecto, las actualizaciones y de qué forma es el versionado. Si va a utilizarse el del Subversion también debe ser descrito.]

Anexo V “Aspectos de infraestructura tecnológica”

<Nombre del Proyecto>

Aspectos de Infraestructura Tecnológica

Versión <#>

Historia de revisiones

Fecha	Versión	Descripción	Autor
<dd/mm/aaaa>	<#>		Responsable (s)

1. Introducción

[Se da una breve introducción de lo que significa la planilla para los proyectos que pretendan hacer su lanzamiento. Debe indicarse la importancia que se le concede a los aspectos que tienen que ver con la infraestructura técnica del proyecto. Se expresa cómo el proyecto va a poner en práctica estos aspectos.]

2. Infraestructura tecnológica.

[Por cada uno de los aspectos que se relacionan a continuación debe relacionar cual es que usado por el proyecto, el objetivo de su uso y las características que va a presentar.]

2.1 Sitios web

[Especificar si tiene el proyecto un sitio web propio de proyecto para la promoción del mismo, si lo tiene; especificar la dirección (url).]

2.2 Wikis

[Especificar que opciones brinda la wikis si se utiliza en el proyecto (subir documentos, intercambio de información.)]

2.3 Canales de comunicación

[Cuáles son los canales por donde los integrantes del proyecto se comunica, por donde hacen la distribución de noticias, de anuncios, etc.]

2.4 Control de versiones

[Especificar cuál es el software que el proyecto utiliza para hacer el control de versiones.]

2.5 Gestión de fallos

[Especificar cuál es el software que el proyecto utiliza para hacer la gestión de fallos.]

2.6 Seguimientos de tareas

[Especificar cuál es el software que el proyecto utiliza para hacer el seguimiento de tareas.]

Anexo VI “Aspectos comunitarios”

<Nombre del Proyecto>

Aspectos Comunitarios

Versión <#>

Historia de revisiones

Fecha	Versión	Descripción	Autor
<dd/mm/aaaa>	<#>		Responsable (s)

1. Introducción

[Se da una breve introducción de lo que significa la planilla para los proyectos que pretendan hacer su lanzamiento. Debe indicarse la importancia que se le concede a los aspectos que tienen que ver con la infraestructura técnica del proyecto. Se expresa cómo el proyecto va a poner en práctica estos aspectos.]

2. Código de conducta

[Establecer las normas de comportamiento dentro del proyecto especificar cómo va a ser el comportamiento dentro de las áreas que abarca el proyecto (foros, chat, correos, reuniones de la comunidad). Además de especificar las acciones que se realizan con los voluntarios en el proyecto (delegación de tareas, reconocimiento individual, bifurcaciones.))]

3. Manejo de voluntarios

[Especificar cuáles son los roles presentes en el proyecto, si se utiliza la delegación de tareas y las acciones de cómo pueden contribuir con el desarrollo del proyecto, ya sean integrantes o no del proyecto, además de cómo dan a conocer esas acciones a la comunidad.]

Anexo VII “Aspectos complementarios”

<Nombre del Proyecto>

Aspectos complementarios

Versión <#>

Historia de revisiones

Fecha	Versión	Descripción	Autor
<dd/mm/aaaa>	<#>		Responsable (s)

1. Introducción

[Se da una breve introducción de lo que significa la planilla para los proyectos que pretendan hacer su lanzamiento. Debe indicarse la importancia que se le concede a los aspectos que tienen que ver con la infraestructura técnica del proyecto. Se expresa cómo el proyecto va a poner en práctica estos aspectos.]

2. Aspectos legales

[En este apartado se deben decir las acciones que realiza el proyecto para aumentar el nivel de conocimientos de los integrantes de acuerdo a la legalidad en el software libre. Se debe decir cuál es la licencia que va a usar el proyecto y si la misma, tiene dualidad con otra, además si se usa el copyleft o no. Se debe especificar si se cuenta con un acuerdo de contribución a la licencia y también a nombre de quién salen las obras, si es individualmente, por el proyecto o por la institución que patrocina este último.]

3. Aspectos financieros

[Especificar quien es el patrocinador del proyecto y qué tipo de financiación es utilizada.]

Glosario de términos

Camino: Camino es un navegador web optimizado para Mac OS X con una interfaz de usuario y potente motor Gecko. Es el simple, seguro y rápido navegador para Mac OS X.

Bugzilla: Bugzilla es un sistema de seguimiento de errores diseñado para ayudar a los equipos de gestión de desarrollo de software. Cientos de organizaciones en todo el mundo están utilizando esta poderosa herramienta para organizarse y comunicarse de manera efectiva.

Firefox: Es el potente navegador web de Mozilla, es rápido, seguro y adaptable a las condiciones del usuario que lo utilice.

Thunderbird: Se 1considera la próxima generación de correo electrónico aportando seguridad, rapidez y facilidad de uso en cuanto al usuario.

Gecko: Motor de diseño capaz de leer contenido web como HTML, CSS, XUL Y Javascript.

XulRunner: Proporciona mecanismos para instalar, actualizar y desinstalar aplicaciones basadas en XUL como Firefox y Thunderbird.

Subversion: Conocido también como SVN es un software para el control de versiones con licencia Apache/BSD.

CVS: Es difundido bajo la licencia GPL y desarrollado por GNU. Permite que los colaboradores trabajen y la ve mantiene un registro de todo el trabajo y los cambios en los ficheros que forman parte del proyecto.

Ssh: Protocolo de comunicaciones donde la transferencia de datos se realiza de forma segura.

Httpd: Activa el servidor web Apache de Linux.

Release: Se refiere al producto final ya preparado para lanzarse como versión definitiva a menos que se encuentre algún error.