Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

# METODOLOGÍA PARA EL ESTUDIO DE IMPLANTACIÓN DE PROGRAMAS DE CÓDIGO ABIERTO EN UNA ORGANIZACIÓN

**Stuardo Enrique Del Aguila Padilla**
Asesorado por el Ing. Everest Darwing Medinilla Rodríguez

Guatemala, octubre de 2006

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA


FACULTAD DE INGENIERÍA

# METODOLOGÍA PARA EL ESTUDIO DE IMPLANTACIÓN DE PROGRAMAS DE CÓDIGO ABIERTO EN UNA ORGANIZACIÓN

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA

FACULTAD DE INGENIERÍA

POR

**STUARDO ENRIQUE DEL AGUILA PADILLA**

ASESORADO POR EL INGENIERO EVEREST DARWING MEDINILLA RODRÍGUEZ

AL CONFERÍRSELE EL TÍTULO DE

**INGENIERO EN CIENCIAS Y SISTEMAS**

GUATEMALA, OCTUBRE DE 2006

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

## NÓMINA DE JUNTA DIRECTIVA

| | |
|---|---|
| DECANO | Ing. Murphy Olympo Paiz Recinos |
| VOCAL I | Inga. Glenda Patricia García Soria |
| VOCAL II | Lic. Amahán Sánchez Álvarez |
| VOCAL III | Ing. Julio David Galicia Celada |
| VOCAL IV | Br. Kenneth Issur Estrada Ruiz |
| VOCAL V | Br. Elisa Yazminda Vides Leiva |
| SECRETARIA | Inga. Marcia Ivonne Véliz Vargas |

## TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

| | |
|---|---|
| DECANO | Ing. Murphy Olympo Paiz Recinos |
| EXAMINADOR | Ing. Cesar Augusto Fernádez Cáceres |
| EXAMINADOR | Ing. Pedro David Tzoc y Tzoc |
| EXAMINADOR | Inga. Virginia Victoria Tala Ayerdi |
| SECRETARIA | Inga. Marcia Ivonne Véliz Vargas |

## HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

# METODOLOGÍA PARA EL ESTUDIO DE IMPLANTACIÓN DE PROGRAMAS DE CÓDIGO ABIERTO EN UNA ORGANIZACIÓN,

tema que me fuera asignado por la Dirección de la Escuela de Ciencias y Sisemas, en septiembre de 2005.

STUARDO ENRIQUE DEL AGUILA PADILLA

# ACTO QUE DEDICO A:

**Dios**

Porque reconozco que todo lo que existe es por su voluntad.

**Mi esposa e hijos**

Por ser mi inspiración en la vida.

**A mis padres**

Por su invaluable apoyo.

**A mi demás familia y amigos.**

# ÍNDICE GENERAL

# ÍNDICE DE ILUSTRACIONES

## FIGURAS



## TABLAS

# GLOSARIO

**Código abierto** Es todo programa o sistema que cumple con los diez criterios definidos por la organización sin fin de lucro llamada *Open Source Foundation*.

**Código fuente** Son los programas escritos por los programadores que se encuentran escritos en un lenguaje de programación y que son leíbles por los seres humanos.

**Distribución** Es el conjunto de uno o más programas, los cuales están listos para ser instalados y tienen asignado un nombre y un número de versión para poder ser identificados y diferenciados de otros.

**Encriptación / encriptado** Se refiere al tratamiento de los datos o archivos para impedir que nadie excepto el destinatario correcto pueda leerlos.

**Internet** Es la red pública de computadoras más grande del mundo que provee muchos servicios para sus usuarios.

**Preprocesador / precompilador** Se refiere a un copilador que reconoce un lenguaje de una generación, el cual lo puede extraer de un programa fuente escrito en otro lenguaje para traducirlo y que el compilador pueda generar el programa ejecutable.

| | |
|---|---|
| **Licencia de programa** | Es el permiso o derecho adquirido, ganado o cedido por parte de los creadores de un programa, dirigido a los usuarios u organizaciones que usan el programa. |
| **Usuario final** | Es toda persona que compra o adquiere de alguna forma un programa para usarlo y así satisfacer sus necesidades informáticas. |

# RESUMEN

En los últimos años ha surgido un movimiento muy interesante y poco comprendido por la mayoría, el cual ha obligado a la industria de la computación ha crear nuevos tipos de licenciamiento, este movimiento es conocido como código abierto y se basa en la aportación de muchos programadores a nivel mundial que donan su tiempo para crear programas y sistemas que compiten en algunas ocasiones con la calidad de sistemas creados por empresas especializadas.

Los objetivos de que la mayoría del código abierto, son la búsqueda del beneficio por medio del desarrollo colaborativo a nivel mundial, ya que cualquier persona puede colaborar con ideas y programando parte de los sistemas. Adicionalmente desean permitir que cualquier persona u organización pueda utilizar los programas o sistemas sin ninguna restricción de uso o costo.

En el presente trabajo de investigación se pudo determinar que las empresas guatemaltecas han visto a estos programas como una alternativa más en el mercado y muchas ya han iniciado a utilizarlo.

El evaluar un programa o sistema de código abierto es muy diferente a hacer una evaluación de programas o sistemas comerciales ya que no existen muchos costos asociados, o los costos están escondidos y adicionalmente se deben tomar en cuenta las particularidades del código abierto. En el presente trabajo de investigación se creó una guía para evaluar este tipo de programas o sistemas.

# OBJETIVOS

## Generales

1. Conocer de las empresas Guatemaltecas que utilizan programas o sistemas de código abierto en sus computadoras, cuales son las factores que se tomaron en cuenta para su utilización.

2. Crear una guía el estudio de implantación de código abierto en una organización.

## Específicos

1. Estudiar el mercado guatemalteco para llegar a conocer si las empresas guatemaltecas usan programas de código abierto en sus computadoras.

2. Identificar el tipo de programas de código abierto que se utiliza.

3. Identificar cuáles son las ventajas y desventajas del código abierto, identificadas por los encargados del centro de informática de las organizaciones guatemaltecas que utilizan código abierto.

4. Desarrollar una guía para la evaluación del uso del software de código abierto.

# INTRODUCCIÓN

En el presente trabajo de graduación se hace una exploración al código abierto para exponer las características principales de estos programas o sistemas y se presenta una reseña histórica para entender porqué este movimiento ha tenido tanto apoyo y aceptación en la industria de la informática.

Debido a que las particularidades de los proyectos de código abierto son muy especiales al momento de tomar una decisión sobre qué programa o sistema puedo utilizar, se exploran dichas particularidades con el afán de luego hacer una evaluación más objetiva.

Debido a que no se conoce la difusión del uso del código abierto en Guatemala y que no es conocida la opinión de los encargados de los sistemas en las organizaciones, se hace un estudio para determinar en las empresas medianas guatemaltecas cuales fueron las razones principales que se tomaron en cuenta en el proceso de evaluación.

Con los la investigación del código abierto y las experiencias de los encargados de sistemas en las organizaciones guatemaltecas, se creó en el último capitulo una guía que puede ser utilizada para evaluar la implantación de un sistema de código abierto en una organización. Esta guía ayuda a buscar y evaluar los programas o sistemas de código abierto para que la organización que tiene una necesidad pueda encontrar el programa correcto, la guía logra hacer el proceso de forma objetiva y por lo tanto asegura obtener una mejor selección del programa o sistema de código abierto.

# 1. INTRODUCCIÓN AL CÓDIGO ABIERTO

## 1.1. ¿Qué es código abierto?

Se le conoce como código abierto a todo programa o fragmento de código que fue creado por un programador o un grupo de programadores, con la finalidad de distribuir el programa para que pueda ser usado libremente, es decir, que no se requiera comprarlo o comprar una licencia de uso. Adicionalmente el programa incluye el código fuente, de tal forma que todas las personas que lo deseen podrán ver como el programa fue construido, incluso podrán hacer las modificaciones que deseen. El objetivo general de estos programadores es hacer que la industria de informática evolucione rápidamente, si es posible, con la colaboración de otros programadores y personas no necesariamente del área de informática, y que no se haga a costa de los altos precios que pagan los usuarios finales.

Una definición más formal dice que el código abierto es todo programa o fragmento de código que cumple con la definición llamada *The Open Source Definition*, la cual esta dividida en 10 criterios o reglas generales que deben ser cumplidas. Si un grupo de personas que crean programas desean distribuirlos como código abierto deben cumplir con estos criterios en su totalidad. Existen grupos de programadores o empresas que distribuyen programas bajo algunos criterios del código abierto, pero en este caso ellos indican cuales criterios son los cumplidos y cuales no. La definición de cada criterio y la explicación de su objetivo principal se encuentra a continuación.

### 1.1.1. Criterio 1: Distribución Libre

"La licencia del programa no debe restringir a nadie de venderlo o entregarlo como un componente de una distribución de otro programa conteniendo partes de múltiples orígenes. La licencia no deberá requerir ningún tipo de regalías u otro tipo de pago por tal venta."

El objetivo de este criterio es hacer que los creadores de programas no puedan evitar que cualquier persona tome los programas o fragmentos de código libre y los distribuya como desee, ya sea igual al original o como parte de otros programas mayores. Como se puede notar no existe ninguna restricción sobre tomar programas de código abierto y venderlos, aunque otros criterios que vienen a continuación imponen algunas condiciones para poder hacerlo legalmente.

### 1.1.2. Criterio 2: Código Fuente

"Los programas deben incluir el código fuente y deben permitir también la distribución del código en formato compilado. Si un programa no es distribuido con su código fuente, debe existir una forma bien documentada de como obtener el código por un costo bajo que cubra solamente su distribución, o bien, por medio de Internet sin cargo alguno. El código fuente debería estar de una forma tal que un programador pueda modificarlo fácilmente, no debe estar encriptado o en una forma intermedia como la salida de un preprocesador o traductor."

Este criterio permite la evolución de los programas ya que se garantiza la distribución del código fuente con los programas, esto es esencial ya que sin

este código no se podrían cambiar los programas para llenar las necesidades de las empresas o personas individuales que los desean utilizar de forma distinta a la pensada por los creadores originales.

### 1.1.3. Criterio 3: Trabajos derivados

"La licencia debe permitir las modificaciones y trabajos derivados de los programas, y debe permitir la distribución de estos nuevos trabajos bajo los términos de la licencia original."

Aquí se define al trabajo derivado como cualquier modificación de un programa o fragmento de código que es distribuido como código abierto y adicionalmente se le permite ser distribuido siempre y cuando sea bajo los términos de la licencia original, esto quiere decir que un programa que nace como código abierto seguirá siendo código abierto a pesar de sus modificaciones.

### 1.1.4. Criterio 4: Integridad del autor del código fuente

"La licencia del programa puede restringir la distribución del código fuente ya modificado, solo si la licencia permite la distribución de archivos de parches del código fuente con el propósito de que el programa sea modificado al momento de ser construido. La licencia debe permitir la distribución del programa construido con modificaciones. La licencia puede requerir que las distribuciones modificadas lleven otro nombre o un número distinto de versión que el programa original."

La idea de este criterio es proteger la reputación de los programadores y los modificadores de programas, así como también, proteger el derecho que tienen las personas que van a utilizar los programas de saber quien construyo el programa, qué modificaciones se le han hecho y quienes tomaron parte en todo este proceso. Este criterio permite que existan limitaciones al modificar el código, pero a la vez deja una salida al indicar que se debe permitir hacer las modificaciones en forma de parches.

### 1.1.5. Criterio 5: No discriminación contra personas o grupos

"La licencia no debe discriminar a ninguna persona o grupo de personas." La idea de este criterio es no permitir la discriminación por motivos humanitarios, así como también, para permitir que el proceso de evolución de los programas pueda llegar a una gran diversidad de personas o grupos que podrían contribuir con el código abierto, la ventaja de la diversidad de culturas es que permite mayor creatividad y por lo tanto una mayor evolución de los programas.

### 1.1.6. Criterio 6: No discriminación contra campos de trabajo

"La licencia no debe restringir a nadie para usar el programa en un campo específico de trabajo. Por ejemplo, no debe restringir que el programa sea usado en un negocio, o en una investigación pura en una universidad."

La intención de este criterio es permitir que cualquier persona individual, empresa u organización de cualquier tipo, pueda usar los programas para uso comercial o no comercial.

### 1.1.7. Criterio 7: Distribución de la licencia

"Los derechos incluidos en el programa deben aplicarse a todos a quienes el programa fue distribuido sin necesidad de ejecutar una licencia adicional entre las dos partes."

El código abierto es distribuido de muchas formas y sin importar como una persona o empresa u organización obtuvo el código, este siempre traerá incluido o adherido uno o más derechos, los cuales normalmente vienen en forma de un documento electrónico donde se explican. Este criterio indica que no es necesario que entre el creador del código y la persona que lo utiliza deba crearse una licencia escrita.

Este criterio nos permite gran flexibilidad ya que de otra forma cada persona que deseara utilizar un programa de código abierto tendría que escribir un contrato, firmarlo y legalizarlo según las leyes del país donde se encuentren.

### 1.1.8. Criterio 8: La licencia no debe ser específica para un producto

"Los derechos incluidos en un programa no deben depender del hecho de que el programa sea parte de una distribución. Si el programa es extraído de la distribución y luego usado o distribuido con los términos de la licencia del programa, todas las partes a las que el programa es distribuido deben tener los mismos derechos como aquellos a los que fueron otorgados por el programa y distribución original."

La idea es que no se pierdan los derechos que el programa original trae, si alguien extrae un programa de una distribución y lo pretende distribuir empaquetándolo como una distribución distinta.

### 1.1.9. Criterio 9: La licencia no debe restringir otros programas

"La licencia no debe poner restricciones a otros programas que son distribuidos con el programa licenciado. Por ejemplo la licencia no puede insistir que otros programas distribuidos en la misma distribución que el programa licenciado también sean programas de *open source*, que sean de un origen determinado, etc."

Este criterio se explica a sí mismo ya que incluye un ejemplo, y básicamente trata de que el código abierto no restrinja nada sobre su distribución con otros programas.

### 1.1.10. Criterio 10: La licencia debe ser tecnológicamente neutral

"Ninguna condición de la licencia puede ser dictada para ninguna tecnología en particular o estilo de interfase gráfica."

Esta condición esta diseñada para todas aquellas licencias que podrían restringir que el programa sea distribuido únicamente por Internet y que requieran un navegador de Internet con una conexión abierta para poder acceder al programa. La idea general es que la licencia no restrinja como puede ser distribuido el programa o qué interfase grafica necesita tener para poder acceder a él.

Los criterios fueron creados por la organización llamada *The Open Source Foundation* y a pesar de que pueden verse como un conjunto complicado de reglas, realmente se trata ordenar todos los conceptos y guiar a los creadores de código abierto para que distribuyan los programas que crean usando las mismas definiciones.

## 1.2. ¿Qué es un proyecto de código abierto?

Un proyecto de código abierto tiene como objetivo construir, mantener y distribuir un programa o un fragmento de programa, cuya licencia cumpla con los 10 criterios de la definición llamada *Open Source Definition*.

En el proyecto de código abierto pueden participar desde un programador, un grupo de programadores, una institución educativa, una organización sin fines de lucro, o bien pueden participar compañías cuyo fin es obtener un lucro a mediano o largo plazo. Por ejemplo una compañía puede ser patrocinador de un proyecto de código abierto y aunque el programa o programas creados se distribuyan libremente, la empresa podría cobrar por dar soporte técnico o por la documentación del programa.

Algunos proyectos de código abierto tienen miembros que deben asociarse primero para participar en todas las actividades del mismo, en otros casos el proyecto lo administra y programa una sola persona.

## 1.3. Certificación de código abierto

Muchos proyectos de código abierto pueden decir que su licencia cumple con los 10 criterios que definen al código abierto, pero podría ser que la licencia sea confusa y no cumpla a cabalidad los 10 criterios. Para evitar estos problemas existe una organización sin fin de lucro llamada *Open Source Initiative* la cual puede recibir las solicitudes de los proyectos para que sean certificados como verdaderos proyectos de código abierto. Esta organización estudia todos los términos de las licencias presentadas y determina si son compatibles con los diez criterios para considerar al proyecto como verdadero código abierto.

La certificación no es obligatoria ni necesaria para ser considerado un proyecto de código abierto, simplemente existe como un mecanismo para que las personas que desean usar código abierto en sus computadoras, se aseguren que la licencia cumple verdaderamente con los criterios definidos para el efecto. Algunas personas podrían sentirse un poco más seguras de poder instalar programas que sí fueron certificados que otros programas que no.

Muchos proyectos pueden no estar certificados, pero no quiere decir que no sean código abierto, lo que podría ocurrir, es que simplemente no les interesa pasar por el proceso de certificación.

## 1.4. Historia del termino “código abierto”

Desde la creación de las primeras computadoras hasta el principio de los años setentas (1970) la creación de programas de todo tipo había sido en un ambiente muy cooperativo, donde muchos profesionales y académicos

cooperaban para crear programas confiables. Al principio de los años ochentas (1980) casi todo el software era propietario, es decir que tenía un dueño quien prohibía y prevenía la cooperación entre los usuarios.

En 1983 se inicia un movimiento llamado *GNU Proyect* con una filosofía que promovía el espíritu de cooperación que existía en los primeros años de la comunidad de informática, y trata de remover los obstáculos de la cooperación. En 1985 el proyecto *GNU* crea la organización llamada *Free Software Foundation* (FSF) dedicada a promover el derecho de los usuarios a usar, estudiar, modificar, copiar y distribuir programas. Esta organización promueve la creación de programas libres conocido en ingles como *free software* y de documentación libre, para que cualquier persona pueda usar los programas y los programadores de todo el mundo puedan modificarlo y contribuir con la evolución del mismo.

El proyecto *GNU* decidió hacer un sistema operativo compatible con *Unix*, pero el sistema operativo realmente necesita mucho software adicional al *kernel*, entonces iniciaron creando compiladores, editores de texto, programas para correo electrónico, etc. Para 1990 el proyecto *GNU* había creado los componentes principales menos el *kernel*, entonces se unió al proyecto *Linux*, el cual es un *kernel* creado por Linus Torvalds y así nació un sistema operativo *GNU* basado en *Linux*. Este acontecimiento es muy importante en la historia ya que marca el inicio de uno de los proyectos de código abierto más exitosos de la historia.

Debido a que muchas personas ya hablan sobre el *GNU Linux* y otros programas que liberaban sus licencias, existía en los años 90 la necesidad de unificar definiciones y conceptos, es así como en 1997 Bruce Parens escribió un documento inicial en sucio llamado *The Debian Free Software Guidelines*, el

cual durante unos meses fue refinando con la ayuda de muchos desarrolladores de software incluyendo los programadores de la distribución *Debian GNU/Linux*. El 3 de febrero de 1998 Todd Anderson, Chris Peterson del instituto Foresigth, John "Maddog" Hall y Larry Augustin de Linux International, Sam Ockman del grupo de usuarios de Linux del *Silicon Valley* y Eric Raymond se reunieron para comentar el anuncio de que la compañía Netscape iba a entregar a todo el mundo el código de su navegador de Internet.

Durante esa reunión Chris Peterson contribuyo con el nombre *Open Source* para denominar a este movimiento que difería un poco del llamado *free software* sobre el cual esta basada toda la filosofía del proyecto *GNU*.

Finalmente en febrero de 1998 se removió la referencia de *Debian* y se creo así la definición de código abierto y la creación de la organización sin fin de lucro llamada *Open Source Initiative*.

## 1.5. Diferencia entre código abierto y programa libres

La motivación de los proyectos de código abierto es un tanto antigua ya que se inició con la creación del proyecto *GNU*, la organización *Free Software Foundation* (FSF) y los términos de las licencias *GNU*. Luego se creo la organización *Open Source Initiative* (OSI) con motivaciones distintas y una filosofía un poco más práctica.

Existen varias diferencias entre programas libres y código abierto, primero, el termino *free software* es ambiguo en su idioma natal, el ingles, ya que una persona puede leer la palabra *free* como gratis, es decir, que no cuesta dinero, entonces esta interpretación puede asociarse a programas que tienen baja

calidad y que por esta razón son gratis. La palabra *free* en el termino *free software* significa: libre, la cual es una palabra política que la FSF utilizo para reafirmar su filosofía. Ellos piensan que los programas deben ser libres y que los programas propietarios son un problema social que debe ser afrontado con programas libres.

En el caso del código abierto, el termino *open source* tiene que ser explicado a través de su definición oficial, no tiene una explicación en una frase simple que enmarque todo su contenido. Si se intenta hacer esto, la definición tendría que tener tantos términos y aclaraciones como su definición oficial. El problema de esto, es que el termino tiende a ser usado por algunas compañías para promover una imagen más atractiva a los consumidores al declarar que sus programas son de código abierto, cuando en realidad no cumplen con todas los criterios para ser considerado como tal. En cambio el termino *free software* tiende a ser más descriptivo y por lo tanto no es común que sea usado para este fin.

Segundo, las licencias de los programas libres son más estrictas respecto a temas como el uso del programa y la documentación, en cambio, el código abierto es más relajado y es por esto que muchas personas lo ha calificado como una metodología de desarrollo y al proyecto *GNU* como un movimiento social. Pareciera ser que la diferencia se encuentra en que el código abierto tiene objetivos más prácticos mientras que el proyecto *GNU* tiene objetivos más idealistas.

Tercero, desde su inicio la organización OSI ha tenido más presencia en los medios noticiosos y ha sido más conocido por las personas que la FSF porque surgió en un momento donde todos están más pendientes de la tecnología, existe más interés en general sobre los problemas del

licenciamiento y el uso del Internet esta más difundido. Seguramente muchas personas ya han oído sobre código abierto pero no han escuchado nada sobre *GNU*.

En general tanto el código abierto como los programas libres son conceptos que benefician al usuario final pero que por sus pequeñas diferencias no tienen los mismos objetivos ni principios, por lo que nunca van a unirse en un solo concepto. Estas organizaciones no compiten entre sí, incluso en algunos proyectos trabajan juntas ya que su objetivo final consisten en desarrollar la industria de la informática a través de desarrollo colaborativo.

## 1.6. Tipos de programas y licencias

Como licencia de programa se entiende a los documentos legales impresos o en formato electrónico que le dan al usuario final derechos o restricciones sobre los programas que dichas licencias protegen. En general la licencia de los programas varia de un programa a otro y podríamos decir que existen tantos tipos de licencias como programas hay en todo el mundo, sin embargo podemos clasificarlas por su orientación y objetivo principal. Al describir el tipo de licencia se puede describir al mismo tiempo el tipo de programa que la licencia protege.

Antes de describir los tipos de licencias, debemos aclarar dos conceptos, uno es el *Copyrigth* que significa que los programas están protegidos y no pueden ser modificados o copiados de ninguna manera, el otro concepto es el *Copyleft* que significa que los programas pueden ser modificados y que la licencia del programa modificado no podrá restringir la libertad del programa, es

decir, tiene que seguir siendo *Copyleft* para que otros puedan seguir modificándolo.

Algunas de los tipos de licencias que se describen a continuación son *Copyrigth* y otras podrán ser *Copyleft* pero no pueden ser ambas.

### 1.6.1. Programas libres

Un programa libre viene con el permiso de usar, copiar, modificar y distribuir el programa ya sea sin modificaciones o con modificaciones y de forma gratuita o cobrada. El código fuente siempre tiene que estar disponible y debe ser distribuido junto con el programa. Este tipo de programa es conocido por el termino en ingles *free software* donde la palabra *free* significa libertad. Este concepto fue creado por el proyecto *GNU* y la licencia utilizada se le conoce como *GNU General Public Licence* (*GNU GPL*).

### 1.6.2. GNU *Lesser Public Licence*

Este tipo de licencia fue creada a partir de la *GNU GPL* que aplica a librerías de código que permiten ser incluidas en cualquier programa, inclusive en los programas propietarios no libres. Este tipo de licencia evita que una librería obligue a que los programas que la utilicen sean considerados trabajos derivados, lo cual podría obligarlos a distribuirse bajo la licencia *GNU GPL*.

### 1.6.3. Código abierto

Básicamente el código abierto es lo mismo que los programas libres, con la diferencia que este tiene criterios menos estrictos referente a la distribución del código fuente y en su contenido se menciona inclusive que el código puede ser vendido con o sin modificaciones siempre y cuando se mencione la licencia del trabajo original.

### 1.6.4. Programas *Copylefted*

Este tipo de programas son programas libres que en sus términos de distribución no permiten que los redistribuidores agreguen restricciones de ningún tipo cuando modifican o redistribuyen el programa. Estos son una especie de programas libres o de código abierto, con la diferencia que son más restrictivos y por lo tanto no son tan populares.

### 1.6.5. Programas no-*Copylefted*

Este tipo de programas vienen con el permiso de los autores de ser modificados, redistribuidos y también de agregar restricciones, es decir que una persona puede tomar un programa *Copylefted* y modificarlo, luego podría escribir una nueva licencia e indicar que el nuevo programa es propietario o que no puede ser redistribuido, o cualquier otra restricción que desee. Este tipo de programas no son muy populares ya que van en contra la filosofía de la mayoría de proyectos de código abierto y programas libres, la cual busca que se sigan desarrollando los programas con el tiempo.

### 1.6.6. Programas semi-libres

Un programa semi libre es todo programa que no es libre, viene con permiso de ser usado, copiado, distribuido y modificado pero para usos no lucrativos. Este tipo de programas básicamente restringe a las empresas y a los gobiernos a utilizarlo, esto está en contra de uno de los criterios principales de los programas libres y del código abierto: la no discriminación.

Normalmente estos programas son distribuidos para que una persona pueda usarlo sin costo en su casa, pero si una empresa o una institución de un gobierno desea utilizarlo, debe de pagar una licencia de uso.

### 1.6.7. Programa propietario

Estos programas no son libres ni semi libres, su uso, redistribución y modificación esta prohibidas, requieren pedir permiso para hacerlo o esta restringido de tal forma que no puede hacerse libremente. La palabra propietario significa simplemente que el programa tiene un dueño que tiene todos los derechos de autor sobre el programa, es decir que es *Copyrigth* y que su uso requiere realizar una compra del programa o de una licencia de uso.

### 1.6.8. Programas de dominio público

Un programa es de domino público cuando no tiene la protección de *Copyright'* ni de *Copyleft*. En algunos casos el código fuente no esta disponible por lo tanto no es programa libre ni código abierto ya que no pueden hacerse modificaciones. Si el código fuente esta disponible al público significa que

algunas copias o modificaciones podrían ser un caso especial de *Copyleft*, por lo que podrían existir copias o modificaciones que no son libres.

Es necesario aclarar que los programas libres y el código abierto generalmente no están en el dominio público ya que tienen *Copyrigth* pero los poseedores del derecho dieron los permisos en su licencia para que los programas puedan ser usados, copiados, modificados y distribuidos libremente, los programas de domino público en cambio no tienen protección alguna.

### 1.6.9. Programas *freeware*

El termino *freeware* es una palabra en el idioma ingles y realmente no tiene traducción ni concepto definido, pero generalmente se usa para indicar que los programas pueden distribuirse libremente, pero no está disponible el código fuente y no puede ser modificado por nadie. Este tipo de programas no son libres, por lo que se recomienda no confundir los términos en ingles de “free software” (programas libres) con *freeware*.

Este tipo de programas tiene dueño y se puede decir que es un programa propietario con la diferencia que puede ser distribuido libremente. Algunos de estos programas tienen un mecanismo que les permite a los creadores hacer dinero sin cobrar licencias, por ejemplo un programa podría usar el Internet para pasar anuncios comerciales en una parte de la pantalla mientras los usuarios hacen uso del programa.

Otros programas *freeware* son versiones más simples de programas propietarios y son usados por sus empresas creadoras para hacer promoción y dar a conocer sus productos, luego si el usuario final ve útil el programa y desea

las características más avanzadas del programa propietario, entonces la empresa le ofrece venderle el programa más completo.

### 1.6.10. Programas *Shareware*

Este tipo de programas no son libres y vienen con el permiso para ser distribuido de cualquier forma, generalmente la licencia indica que una persona puede probarlo por un período de tiempo, pero si luego de probarlo desea continuar su uso, es necesario pagar el costo de su licencia. En algunos casos este tipo de programas impone la restricción del uso solamente en los términos de su licencia y en otros casos el programa tiene programado un mecanismo que hace que el programa ya no funcione después de un determinado tiempo o al llegar a una fecha específica.

Este tipo de programas son propietarios con una forma de distribución un poco más flexible, pero al final siempre buscan que los usuarios finales paguen por el uso del programa.

### 1.6.11. Programas comerciales

Este tipo de programa es creado por una empresa con el propósito de hacer dinero con el uso de él. Programas comerciales no son lo mismo que programas propietarios aunque la mayoría de programas propietarios son comerciales ya que se venden, ahora bien también pueden existir programas comerciales libres por ejemplo cuando alguien toma un programa libre y luego vende contratos de soporte técnico. En este ejemplo los clientes de la empresa tienen el programa de forma libre ya que no pagaron por él, inclusive podrían hacerle modificaciones, pero mantienen una relación comercial con la empresa

que les vende el soporte técnico y probablemente mantenimiento del programa. También puede existir programas no comerciales no libres, como en el caso de alguna universidad que crea un programa en un proyecto de investigación pura, donde el programa final va a ser vendido y no es permitido distribuirlo fuera de dicha universidad.

## 1.7. Implicaciones legales

Tanto el código abierto como los programas libres tienen licencias, tal y como las tienen los programa propietarios comerciales, la diferencia está en que sus licencias dan permisos específicos a las personas que desean utilizarlos. Es responsabilidad de la persona u organización que va a utilizar estos programas de leer los términos de la licencia y así determinar si el uso, la copia, modificación o distribución que desean hacer está contemplada dentro de la licencia y la hace una acción legal.

Se debe tomar en cuenta que el código abierto impone menos restricciones que el programa libre, y que muchos utilizan el machote de este tipo de licencias para distribuir sus programas, pero otros construyen licencias muy específicas para sus programas y que podrían, o no, ser compatibles en un 100% con las especificaciones del código abierto o programa libre.

El código abierto es creado por una persona o por un grupo de personas que luego permiten su distribución de forma abierta, y para esto utilizan los términos de las licencias. Como el objetivo principal es que participen dentro del proyecto tantos programadores como sea posible, muchos proyectos han tenido que crear una especie de comité o grupo especial de decide cual será la dirección del proyecto y que mejoras se realizarán en base las peticiones

recibidas. Normalmente ocurre cuando los proyectos se vuelven muy grandes y por lo mismo son más difíciles de manejar.

Legalmente las personas que dirigen los proyectos o crean los programas son los que tienen los derechos de autor, quienes a su vez dan permisos especiales para el uso, copia, modificación y distribución de los programas creados dentro del proyecto. Esto debe ser así para evitar que cualquier persona tome el código abierto y proclame que es suyo y con él empiece a realizar acciones contrarias al espíritu de libertad que el código abierto y los programas libres promueven. El otro objetivo es que exista un orden y organización para determinar que cambios son necesarios, quién lo realizará y luego unir todos los trabajos individuales de los programadores para distribuir las nuevas versiones de los programas.

Se explicará a continuación las implicaciones legales que podrían existir al usar, copiar, modificar y distribuir código abierto.

### 1.7.1. Uso

El uso del código abierto o de programas libres no posee ninguna implicación legal adversa a la persona u organización que lo use ya que este es el objetivo principal de todos estos proyectos.

### 1.7.2. Copia

Las licencias permiten realizar copias de los programas para uso propio, como copia de respaldo o para ejercer el otro derecho, el cual es el de distribución. Muchos programas propietarios y comerciales permiten hacer

copias solamente como copia de respaldo y otros no permiten hacer ningún tipo de copia, esto es una gran desventaja y es la base de todo programa propietario.

Por lo anterior, podemos concluir que para un programa de código abierto o libre, no existe ninguna implicación legal adversa hacia nosotros si realizamos copias de cualquier tipo de dicho programa.

### 1.7.3. Modificación

Este derecho es un poco más complejo de ejercer por un usuario común y corriente, normalmente las modificaciones las hacen programadores que conocen muchos aspectos técnicos implicados dentro de los proyectos, entre ellos el lenguaje de programación utilizado, la arquitectura, el diseño de las estructuras, el ambiente al cual esta destinado, etc.

Si una persona individual o una organización realiza modificaciones debe hacerlo bajo el termino de la licencia, el cual normalmente indica que se debe de mencionar el origen del trabajo inicial, es decir, el proyecto de código abierto o programa libres sobre el cual se basó. Existen dos motivos para los cuales una persona individual o una organización decida realizar modificaciones: para uso propio o para crear un trabajo derivado.

Si la modificación es para uso propio no existe mayor implicación ya que solamente se debe mencionar en la licencia que el programa que estamos usando esta basado en el proyecto de código abierto o programa libre.

Si la modificación es para crear un trabajo derivado, entonces debemos mencionar el origen del proyecto y que el nuevo proyecto es un trabajo derivado distinto al proyecto original. Aquí debemos tomar en cuenta que algunas licencias obligan a que los trabajos derivados sigan siendo proyectos de código abierto o programa libre, pero otras no. Si no se cumple con la licencia original, nuestro trabajo derivado podría estar infringiendo la ley y los administradores del proyecto original podrían tomar acciones legales contra los infractores.

### 1.7.4. Distribución

Este es el tema más polémico cuando se habla de programas propietarios y comerciales, ya que en ese caso el objetivo principal de hacer el programa comercial es para obtener ganancias al vender del trabajo realizado. Durante toda la historia de la informática los programas propietarios y comerciales han luchado para evitar la distribución no autorizada de sus productos. En el caso de proyectos de código abierto o de programas libres es totalmente lo contrario ya que ellos buscan que sus proyectos sean distribuidos por todos los medios posibles para fomentar su uso.

Si una persona individual o una organización desea distribuir un proyecto de código abierto o programa libre, puede hacerlo sin ningún problema, solamente hay que tomar en cuenta que en algunos casos no podremos hacer una distribución por medio de una página *web* a menos que pidamos autorización a los administradores del proyecto. En otros casos podemos distribuirlo en una página web, pero no podremos decir que somos un distribuidor oficial a menos que pidamos autorización.

Hay que tomar en cuenta que algunos trabajos derivados tienen la restricción de ser distribuidos bajo los términos de la misma licencia que el proyecto original.

Muchos personas tienen la duda sobre si es legal cobrar dinero por el código abierto o por programas libres. Para explicar esto tomaremos como ejemplo a la compañía Red Hat Inc. la cual distribuye el sistema operativo llamado GNU/Linux, el cual es más conocido simplemente por Linux. Esta compañía toma al sistema operativo, crea un instalador amigable, toma otros proyectos open source o programas libres como por ejemplo: editores de texto, programas para oír música, hojas de calculo, etc. y todo esto lo junta en una distribución llamada Red Hat Linux. La distribución es cobrada ya que la compañía ofrece un disco compacto (*cd*) con un instalador especial de la distribución, además entrega un libro el cual es un manual de instalación y configuración. Adicionalmente la empresa suele cobrar por soporte técnico para la instalación y el uso del sistema operativo.

En este ejemplo podemos ver que la compañía Red Hat no está cobrando por el sistema operativo, está cobrando por los costos de la creación de los discos compactos, por la creación del manual de instalación y por el soporte técnico. Como la distribución llamada Red Hat Linux esta siendo distribuida con los mismos términos del proyecto Linux, entonces uno puede tomar un disco compacto de la distribución y crear múltiples copias, distribuirlos entre las personas que deseamos e inclusive se podría cobrar por estos discos copiados.

Es legal y moral cobrar por estas distribuciones, ya que el proyecto Linux seguiría siendo libre, es decir, las personas a las que alguien le venda una copia de los discos compactos pueden usar, copiar, modificar y distribuir el sistema operativo usando la licencia original, pero se les debe aclarar que el

Linux puede ser obtenido gratuitamente desde una página web, obtenido por medio de la compañía Red Hat, o por medio de otros canales de distribución como Suse, Mandrake, etc. y lo que se está cobrando realmente es el servicio de tomar discos en blanco y hacer las copias.

Los trabajos derivados que son creados como por ejemplo el instalador de la distribución Red Hat, también se distribuyen bajo los términos de código abierto, por lo tanto es legal y moral crear copias de estas distribuciones y distribuirlas como más nos parezca.

## 1.8. Los proyectos de código abierto y programas libres más conocidos

Desde la creación del concepto de programa libre y luego el concepto de código abierto se han creado muchos proyectos pero, algunos han sobresalido más que otros debido en algunos casos a su calidad y en otros por una rápida adopción de los mismos. A continuación se mencionarán a los proyectos más conocidos separados por área de aplicación.

En el caso de los sistemas operativos el proyecto que indudablemente ha revolucionado la industria de la informática es GNU/Linux que normalmente se le conoce simplemente como Linux. Este surgió por la unión del *kernel* programado principalmente por Linus Torvalds, y los demás componentes necesarios para su funcionamiento creados por el proyecto GNU. Luego de su unión, empezaron a surgir múltiples distribuciones las cuales simplifican el proceso de instalación y configuración del mismo. En la actualidad el sistema ha probado ser tan eficiente como los sistemas operativos comerciales y en

algunos casos hasta con mejor rendimiento. Otros sistemas operativos libres o de código abierto muy utilizados son FreeBSD, OpenBSD y NetBSD, todos ellos basados en la distribución de Unix de Berkeley desarrollado en la Universidad de California. Otro sistema basado muy conocido basado en el Unix de Berkeley es Darwin el cual es la base para el sistema Mac OS X de Apple Computers Inc..

Entre los servidores de Internet tenemos al proyecto *Apache Web Server*, el cual corre en la actualidad más del 60%, hasta la fecha, de todos los sitios de Internet. Este servidor de páginas de Internet es tan eficiente y fácil de extender que muchas compañías que venden programas comerciales lo han incluido en sus productos tal y como lo hizo IBM y Oracle, las dos compañías más grandes de software empresarial en el mundo.

En el área de redes de computadoras, el programa BIND es el que provee el servicio de *Domain Name Service* para todo el Internet. Este sistema fue adoptado al inicio de la Internet y debido a su proliferación no ha existido ningún programa comercial que haya podido reemplazarlo.

Para el correo electrónico el programa Sendmail es el más utilizado en el mundo para el transporte y ruteo del correo electrónico. En la actualidad prácticamente todos los sistemas operativos basados en Unix utilizan el Sendmail en sus distribuciones comerciales.

Para navegar en Internet existe el navegador llamado Mozilla, el cual es un rediseño del antiguo Netscape y está disponible para casi todas las plataformas desde Linux, la mayoría de tipos de Unix y MS Windows.

Entre los servidores tipo proxy se encuentra el Squid el cual tiene un rendimiento incomparable y tiene muchas opciones de integración con sistemas de autenticación y autorización.

Entre los servidores de aplicaciones J2EE existen dos grandes proyectos, uno es Jboss el cual está 100% construido en java lo cual permite su portabilidad a cualquier plataforma y Tomcat el cual es un proyecto de la organización Apache que tiene un motor de *servlets* y páginas *JSP*. Ambos son excelentes productos y en muchos casos las organizaciones los han seleccionado en lugar de usar productos comerciales.

En el área de automatización de oficinas existe el Open Office el cual es un conjunto de programas que están interconectados para ofrecer una solución completa para casi cualquier organización. El Open Office consta de una hoja de calculo, un procesador de palabras, un creador de presentaciones, un editor de imágenes y una pequeña base de datos. Esta herramienta esta ganando más adeptos cada día ya que ofrece compatibilidad con los archivos equivalentes a los programas de la compañía Microsoft, que actualmente posee acaparada la mayor parte del mercado de automatización de oficinas.

# 2. CARACTERÍSTICAS DE LOS PROGRAMAS DE CÓDIGO ABIERTO

## 2.1. Objetivos de los proyectos

La característica primordial de los proyectos de código abierto es que sus objetivos son orientados a crear una alternativa que sea económica -ya que no se pagaría por los programas al querer usarlos- y de alta calidad para que pueda competir con los programas comerciales.

Desde el punto de vista del proceso de desarrollo los objetivos de crear un proyecto de código abierto son: primero, desarrollar código entre muchos programadores y colaboradores para acelerar el proceso de creación del programa. Segundo, acelerar el proceso de aseguramiento de calidad ya que muchas personas prueban los programas encuentran fallas y algunos casos hasta sugieren la solución.

## 2.2. Integrantes del proyecto

Los integrantes del proyecto no son solamente los programadores que escriben el código, son todas las personas que participan de una forma u otra dentro del proyecto. Existen proyectos donde una sola persona se encarga de analizar, diseñar, construir y hacer pruebas sobre los programas creados, pero generalmente son grupos de personas los que crean un proyecto.

En algunos casos el grupo de personas pertenece a una entidad educativa como por ejemplo una universidad, en otros casos los proyectos son administrados por organizaciones sin fin de lucro que reciben donaciones para poder costear sus gastos.

También existen empresas patrocinadoras que pueden donar equipo, material de oficina, experiencia tecnológica o dinero para que un proyecto de código abierto pueda existir. Otras empresas han transferido programas comerciales que originalmente vendían a código abierto, esto se hace por medio de un proceso que denominan donación, donde la empresa donadora selecciona una organización a la cual le entrega los programas para que de allí en adelante se administren como código abierto. Otras han creado proyectos de código abierto que ellas mismas administran y que han prometido seguir administrando, tal es el caso de la corporación Oracle que recientemente creo el producto *Cluster File System* para Linux y lo distribuye como código abierto. El objetivo de este producto es proveer la base que necesitaba su producto líder, el cual es la base de datos relacional Oracle 10g, para que pueda funcionar en ambientes de cluster sobre Linux. El aporte fue bien recibido por la comunidad de Linux ya que al ser código abierto, toda persona que desea utilizar este producto puede hacerlo, aunque no sea para la base de datos Oracle 10g.

## 2.3. Motivación de los programadores

Al preguntarle a distintos programadores las razones por las cuales apoya un proyecto de código abierto y en algunos casos utiliza su tiempo libre para crear programas que son distribuidos libremente, se obtendrían tantas

respuestas como programadores existen en el mundo, pero en general las respuestas más comunes son:

1. Crear programas que puedan ser utilizados por muchas personas sin que tenga que pagar grandes sumas de dinero por concepto de licencias.
2. Beneficiarse por el desarrollo comunitario para alcanzar dos objetivos: mejorar la calidad de los programas y disminuir tiempo necesario para crearlos.
3. Crear una alternativa que se pueda tomar en cuenta.

Muchos autores de artículos de revistas, comentaristas, reporteros y académicos concuerdan que estas respuestas son las oficiales, pero que realmente existen motivos un poco más grandes y complejos que estos. Algunas personas incluso atacan a los integrantes de los proyectos indicando que lo que buscan es reconocimiento público, o bien, obtener un mejor trabajo en otra empresa diferente a la que laboran actualmente.

Al respecto podemos decir que muchos de los integrantes de los proyectos solamente son conocidos por sobrenombres, algunos inclusive son contribuidores anónimos. Otro factor importante es que no solamente individuos colaboran en el proceso, sino también instituciones como universidades y empresas privadas.

Si asumimos que los programadores quieren ganar reputación y obtener oportunidades de trabajo, entonces solamente existirían programadores que quisieran trabajar en proyectos grandes y mundialmente reconocidos. Además existirían muchos cambios de programadores de un proyecto a otro, asumiendo que un programador podría ver a otro proyecto como un escalón más que alcanzar, y que al estar arriba del proyecto donde esta involucrado actualmente,

tendría un mayor reconocimiento y ganaría reputación. En la realidad los cambios de programadores en un proyecto se ven muy poco y también rara vez alguien abandona un proyecto en forma definitiva.

Después de analizar todo lo que se dice sobre los programadores de código abierto, pareciera que programadores están en contra grandes corporaciones que venden programas y de la presión que ejercen sobre todas las empresas y personas individuales que usan computadoras, al amenazar con auditorias exhaustivas y a obligar a pagar licencias de programas que ya se estén utilizando.

El sentimiento que buscan los programadores es la libertad, la cual se puede alcanzar al utilizar los programas. También hay que aceptar que algunos de los programadores buscan ser parte de la historia de la informática al producir cambios sustanciales en la industria. Estos programadores ya no desean que empresas como Microsoft impongan la manera en que las personas deben trabajar y qué estándares debemos aceptar. El estilo de Microsoft de manejar su mercado y de utilizar prácticas que muchos consideran monopólicas contra sus oponentes, hace que exista más que un buen pretexto para que un programador quiera ser parte de proyectos abiertos donde todos pueden proponer cambios, encontrar problemas y colaborar con la solución.

Existen muchos proyectos que fueron creados para pelear directamente contra productos de Microsoft, por ejemplo Linux ya ha rebasado por mucho al sistema operativo Windows NT ó 2000 en áreas como estabilidad, escalabilidad, desempeño y por supuesto en seguridad. También existen proyectos que no tienen un rival directo de la empresa Microsoft, pero que tienen el mismo espíritu de libertad y desarrollo colaborativo que hacen estos proyectos puedan surgir exitosamente.

## 2.4. Actividad en el proyecto

La actividad se refiere a la cantidad de cambios que los programadores hacen sobre el código de los programas. Existen proyectos que están constantemente cambiando y produciendo nuevas versiones de los programas en pocos meses. También existen proyectos que cambian muy poco y otros que pueden anunciar que no habrán cambios por un tiempo determinado.

Esto es importante saberlo ya que si el proyecto tiene mucha actividad y si se encuentra una falla en un programa, se puede concluir que al notificarla hay una gran probabilidad de que en poco tiempo exista una corrección o una nueva versión que ya no tenga la falla. Si existiera poca actividad la probabilidad de encontrar una solución al problema se reduce grandemente.

## 2.5. Calidad del producto final

Según su definición “la calidad consiste en que tan bien cumple un producto o servicio el objetivo que se supone debe cumplir – qué tan estrecha y confiablemente satisface las especificaciones para las que se elaboró o proporcionó”.

Si se toma la definición anterior de calidad, se puede evaluar fácilmente si un programa es de alta, mediana o de baja calidad, siempre y cuando sea posible determinar si los programas cumplen con los objetivos trazados. Como los objetivos ya están definidos desde el inicio del proyecto de código abierto, es muy simple poder determinar la calidad de los programas, para esto se

deben realizar dos pasos: primero, se debe ver la documentación del programa y segundo se deben realizar pruebas de los programas en la computadora.

## 2.6. Documentación

La documentación es primordial en cualquier programa ya que sin ella es como manejar un carro con los ojos cerrados. En los proyectos de código abierto existen dos tipos de documentación, una es la documentación interna del proyecto que normalmente sirve para que los programadores puedan realizar cambios en el código, en los lugares adecuados, y la segunda es la documentación orientada al usuario final, la cual la comprenden los manuales de usuario final, manuales de referencia, tutoriales, ayuda en línea, etc.

En algunos casos la documentación del usuario final es distribuida con los programas, pero en otros casos la documentación debe ser accedida en una página web para su uso en línea o como un archivo que debe ser descargado por cualquier persona.

Debe tomarse en cuenta que algunos proyectos no producen su propia documentación y existen proyectos paralelos con distintos miembros cuyo objetivo es crear la documentación para el proyecto principal.

Existen proyectos cuya documentación esta en planes de implementación y en otros casos personas particulares u organizaciones crean la documentación pero cobran por ella. Estos casos son muy particulares y existen muy pocos.

## 2.7. Soporte técnico

El soporte técnico es la ayuda que se puede recibir por parte de expertos cuando se encuentra un problema en los programas que estamos utilizando. El soporte técnico no esta disponible para todos los proyectos y generalmente es proporcionado por correo electrónico, foros en Internet, grupo de noticias, *IRC chat* o por teléfono.

En la mayoría de casos cuando una persona pide soporte técnico, existe una persona experta que tiene que llevar un seguimiento sobre el problema y documentarlo para que en futuras ocasiones sea más fácil resolverlo. Realizar este trabajo significa que se requiere tiempo y recursos por lo que algunas compañías surgieron para dar un servicio de soporte de alta calidad para proyectos de código abierto, pero este servicios es cobrado.

El hecho de que algunas compañías cobran el soporte técnico de programas de código abierto no hace menos atractivo su uso, al contrario lo hace más factible para las organizaciones que requieren un soporte profesional para poder implantar un programa o sistema. En algunos casos las organizaciones tienen políticas que indican que se debe contratar soporte técnico para todos los programas o sistemas que utilizan.

## 2.8. *Bugs* y parches

Se conoce como *bug* a un comportamiento no esperado de un programa o un comportamiento diferente a lo documentado. No es necesario que se despliegue un error para que un comportamiento sea considerado un problema. Los *bugs* existen porque los programas son creados por seres humanos que

eventualmente cometen errores de programación, o bien, no consideraron todas las posibles alternativas durante la etapa construcción del programa.

Las soluciones de los problemas de programación o *bugs* son corregidos por los programadores del proyecto y no necesariamente por las personas que le dan seguimiento a los problemas. Luego corregir un programa existen dos formas de distribuir la solución a un *bug*: 1) se vuelve a distribuir el programa o sistema completo, pero con un número de versión más alto. 2) se puede distribuir solamente la parte del código que sufrió cambios, a esta distribución se le conoce como parche. Los parches tiene como objetivo hacer más rápida la distribución y la instalación de la solución en sistemas o programas que se encuentran en producción. Los parches también suben el número de la versión de los programas o sistemas. No todos los proyectos de código abierto crean parches, algunos tienen la política de esperar hasta la próxima publicación de la siguiente versión. Esto puede retrazar la compostura de los *bugs*, pero es más simple de administrar.

Los número de versión sirven para llevar un control sobre los cambios que se realizan sobre los programas, de tal forma se pueda saber si un programa instalado es susceptible a un *bug*, ya que podría estar documentado que el *bug* ocurre en un número de versión específica.

## 2.9. Mejoras o nuevas características

En algunos proyectos de código abierto existen mecanismos para que los usuarios puedan hacer sugerencias sobre las mejoras que los programas deben de tener en las futuras versiones. No todos los proyectos permiten las sugerencias con un mecanismo preestablecido, pero la mayoría tienen una

forma de contactar a los directores o programadores del proyecto, para sugerir nuevas características en los programas.

## 2.10. Seguridad Informática

Se entiende como seguridad informática a los mecanismos de autenticación y autorización que los programas utilizan para permitir solamente el acceso autorizado de los usuarios a los programas y datos. La seguridad es muy importante ya que evita que intrusos, saboteadores o incluso virus informáticos comprometan los sistemas.

Es muy importante investigar si los programas que se desean utilizar tienen fallas de seguridad, normalmente conocidas como vulnerabilidades. Existen sitios de Internet especializados en el tema de seguridad donde se puede buscar información sobre el programa o sistema de código abierto que se investiga. El sitio más conocido es el de la organización CERT (www.cert.org) el cual mantiene al día los anuncios de las vulnerabilidades de todos los sistemas en general.

La mayoría de proyectos de código abierto son seleccionados por muchas organizaciones porque proveen igual o mejor seguridad que muchos productos comerciales.

## 2.11. Rendimiento

En el contexto de sistemas de información el rendimiento se refiere a la capacidad de un programa o un sistema de procesar información, es decir, un

programa tiene más rendimiento si puede procesar más información en la misma cantidad de tiempo que otro programa, en condiciones iguales.

Existen algunas organizaciones como la TCPM que establece una medida estándar de transacciones por minuto para calcular el rendimiento de bases de datos. Para otro tipo de sistemas existen otro tipo de medidas, por lo que se debe contactar a los creadores del proyecto de código abierto para determinar cuál es el estándar que debe usarse para este propósito.

Es muy importante investigar las medidas de rendimiento, pero también investigar el ambiente donde se realizaron las pruebas, ya que en algunos casos los creadores de los proyectos solamente publican las medidas sobre los sistemas donde se tiene buen rendimiento y no publican las medidas donde se produce mal rendimiento.

En general los programas de código abierto no producen mejor rendimiento que los programas comerciales, pero en algunos casos el rendimiento si es mejor y es por esta razón que, algunas compañías como IBM y Oracle distribuyen algunos programas de código abierto junto a sus productos comerciales.

## 2.12. Cómo obtener programas libres o de código abierto

Los encargados de los proyectos de código abierto y programas libres siempre intentan llegar a la mayor cantidad posible de personas para incentivar el uso de sus programas. La forma más simple ha sido utilizar las redes de computadoras publicas o privadas que han surgido a través de la historia. Al inicio de Internet los proyectos distribuían sus programas utilizando servicios de

noticias y servidores de *FTP* (siglas en inglés para Protocolo de transferencia de archivos), los cuales eran accedidos por medio de interfases carácter y líneas de comandos.

Cuando fue surgiendo el nuevo servicio de *World Wide Web* los proyectos empezaron a escribir páginas en *HTML* que describían las características de los programas y enlaces para descargar los archivos necesarios para instalarlos. La forma más común de buscar los proyectos era utilizando motores de búsqueda como Yahoo, Alta Vista, InfoSeek o más recientemente Google.

En la actualidad la forma más simple de buscarlos es entrar a los sitios especializados en código abierto. Estos sitios proveen a los desarrolladores y a los administradores de los proyectos un conjunto de herramientas que permiten llevar control de versiones, publicar documentación, reportar *bugs*, crear foros de discusión y publicar los programas para que puedan ser bajados a las computadoras de las personas que los necesitan. Estos sitios especializados proveen un motor de búsqueda y llevan estadísticas de los proyectos más accedidos.

Cualquier persona o grupo puede crear en estos sitios un proyecto y pueden aprovechar toda la infraestructura que proveen sin costo alguno. Los sitios generalmente sobreviven por la venta de publicidad en sus páginas y la venta de servicios adicionales vendidos en forma de subscripciones anuales. Los sitios de este tipo más conocidos son Source Forge (http://www.sourceforge.net) y Fresh Meat (http://freshmeat.net).

Hay que tomar en cuenta que en el caso de Fresh Meat las empresas con programas propietarios que son vendidos también pueden publicar la descripción de sus sistemas, aunque no sean de código abierto. Esto hace que

la búsqueda en este sitio sea un poco más difícil ya que hay que estar leyendo cuidadosamente los tipos de licencia para determinar si un programa es de código abierto o si es propietario.

En el caso de *Source Forge*, no ocurre esto ya que ellos solamente admiten el registro de proyectos de código abierto o programas libres.

# 3. ESTUDIO DE ORGANIZACIONES GUATEMALTECAS

Se diseñó una encuesta orientada a las organizaciones guatemaltecas que tienen un departamento de informática con por lo menos un encargado o gerente de informática y un empleado adicional. El objetivo de la encuesta es obtener información sobre el uso del código abierto en las organizaciones guatemaltecas, los aspectos que las organizaciones tomaron en cuenta al implantar un programa o sistema de código abierto, el tipo de programas que usan y los resultados obtenidos.

## 3.1. Objetivos Específicos

1. Llegar a conocer las razones por las cuales empresas Guatemaltecas y organizaciones gubernamentales usan o no programas de código abierto.

2. Llegar a conocer la cantidad de empresas y organizaciones que usan programas y sistemas de código abierto.

3. Llegar a conocer las áreas de la informática donde se están utilizando programas o sistemas de código abierto.

4. Llegar a conocer los aspectos que se tomaron en cuenta para tomar la decisión de adoptar los programas de código abierto.

5. Llegar a conocer qué problemas existieron durante la implantación.

## 3.2. Universo y Muestra

El presente trabajo final de graduación no trata de demostrar las razones por las cuales se usan o no, los programas de código abierto. Este trabajo se enfocará en obtener la información sobre los aspectos que se tomaron en cuenta para tomar las decisiones de adoptar un programa de código abierto y los problemas que existieron para su implantación, para lograr así crear una mejor guía de evaluación para la implantación de programas de código abierto, la cual se explicará en el siguiente capitulo.

El universo de la encuesta es toda organización guatemalteca que tienen un departamento de informática con por lo menos un jefe o gerente de informática y un empleado adicional. No se toman en cuenta las organizaciones que no tienen departamento de informática, ya que generalmente ellas resuelven sus problemas de informática apoyándose en consultores o empresas consultoras externas. En este caso el uso del código abierto dependería de la opinión de una persona y no de un proceso de evaluación como ocurre cuando hay un gerente de informática y su departamento tiene uno o más empleados.

La muestra puede ser pequeña ya que buscamos información para crear nuestra guía de implantación y no estamos demostrando una tendencia.

## 3.3. Diseño del Cuestionario

El cuestionario debe estar escrito de forma clara y debe tener algunas características especiales para que los gerentes de informática respondan las preguntas con total libertad y comodidad. Las características especiales son explicadas a continuación.

### 3.3.1. Anonimato

La encuesta será anónima ya que no se preguntará el nombre de la persona encuestada ni el nombre de la organización donde trabaja. La información que se necesita sobre la organización es su tipo, es decir, si es una organización privada, del gobierno, transnacional o multinacional.

El anonimato provocará que las personas contesten todas las preguntas sin preocuparse de revelar información importante sobre sus sistemas. Algunas organizaciones tienen como política no dar a conocer información a nadie para evitar problemas de seguridad.

### 3.3.2. Tipos de preguntas

La mayoría de preguntas serán de selección múltiple donde los encuestados pueden marcar una o varias de las posibles respuestas, según el caso. En algunas de las preguntas también se podrán seleccionar una respuesta llamada “otros” la cual tendrá un espacio en blanco a su derecha para que puedan agregar una respuesta que no este enumerada.

También existirán preguntas directas las cuales servirán para que describan respuestas más generales, por ejemplo ¿Que recomendaciones puede dar para la implantación de código abierto?.

### 3.3.3. Orden de las preguntas

Para que sea más fácil contestar la encuesta, las preguntas serán agrupadas según la información que se desea obtener. Los grupos de preguntas son: identificación del tipo de organización, conocimientos generales de código abierto, uso de código abierto e implantación de programas.

### 3.3.4. Redacción de las preguntas

Las preguntas fueron redactadas con palabras sencillas y de tal forma que las respuestas sean dadas con exactitud. Las preguntas fueron redactadas de una forma neutral para que la persona que la esta respondiendo no perciba que la encuesta esta a favor del código abierto o en contra.

## 4. Cuestionario

Estudio de la utilización de código abierto
y programas Libres

La presente encuesta tiene como objetivo realizar un estudio sobre el uso del código abierto y los programas libres en las organizaciones guatemaltecas y los factores analizados para su implantación. Las respuestas serán utilizadas únicamente para fines académicos.

1. ¿En qué tipo de organización trabaja? (marque una)
   - ☐ Empresa Privada Nacional
   - ☐ Empresa Transnacional
   - ☐ Empresa Multinacional
   - ☐ Organización sin fines de lucro
   - ☐ Institución educativa
   - ☐ Gobierno / Organización gubernamental
   - ☐ Otra, especifique:

2. ¿Conoce los términos “código abierto” y “programa libre”? (marque una)
   - ☐ Si
   - ☐ No

3. ¿En su organización utilizan programas o sistemas de código abierto o programas libres? (marque una)

☐ Si (pase a la pregunta 5)

☐ No (conteste la pregunta 4 y allí finaliza la encuesta)

4. ¿Cuáles son los motivos principales por los que su organización no utiliza programas libres o de código abierto? (marque los que apliquen)

☐ Difícil implantación
☐ Pocas opciones existentes
☐ Mal rendimiento
☐ No son seguros (seguridad informática)
☐ Los programas/sistemas son dificiles de usar para el usuario final
☐ Porque se pueden modificar y los programadores pueden hacer mal uso de esto
☐ No tienen o tienen poco soporte
☐ Por políticas de la organización
☐ No tienen o tienen poca documentación
☐ No tienen plan de mejoras
☐ No existe un responsable directo
☐ No tienen plan de mantenimiento con parches
☐ No tienen plan de mantenimiento con parches
☐ No existen aplicaciones financieras, contables, ERP, CRM, etc.
☐ Otros, especifique :

5. ¿En cuáles áreas se utilizan estos programas? (marque las que apliquen)

☐ Bases de datos
☐ Office Suite
☐ CAD/CAM
☐ Router
☐ Diseño de páginas Web
☐ Servidor de correo

- ☐ Educación /Entrenamiento
- ☐ Firewall
- ☐ Lenguajes de programación
- ☐ Multimedia / Animación
- ☐ Navegador de Internet
- ☐ Otros, especifique:
- ☐ Servidor de proxy
- ☐ Servidor de web
- ☐ Servidor DNS
- ☐ Sistema operativo para PC
- ☐ Sistema operativo para servidor

6. ¿Cuáles son los motivos principales por los que su organización utiliza programas libres o de código abierto? (marque las que apliquen)
   - ☐ Fácil implantación
   - ☐ Única opción existente
   - ☐ Fácil de usar
   - ☐ Porque se pueden modificar
   - ☐ Tiene soporte
   - ☐ Más seguro que programas
   - ☐ Otros, especifique:
   - ☐ Bajo costo
   - ☐ Alto rendimiento del programa
   - ☐ Se evitan problemas de licencias
   - ☐ Fácil de obtener
   - ☐ Mejor documentación

7. ¿A cuáles aspectos se les dio más importancia dentro de su organización cuando se decidió implantar un programa libre o de código abierto? (marque todas las que apliquen)
   - ☐ Seguridad Informática
   - ☐ Documentación
   - ☐ Licenciamiento
   - ☐ Otros, especifique:
   - ☐ Soporte
   - ☐ Rendimiento
   - ☐ Bugs, Parches

8. ¿Qué problemas encontró al implantar un nuevo programa libre o programa de código abierto? (marque todas las que apliquen)

- ☐ Resistencia de los usuarios al cambio
- ☐ Nada/poca documentación
- ☐ Nada/poco soporte
- ☐ Problemas de instalación
- ☐ Curva de aprendizaje muy grande
- ☐ Problemas internos (bugs)
- ☐ Otros, especifique:

9. ¿Sabía Ud. que algunas compañías como IBM y Oracle han incorporado productos de código abierto dentro de sus productos comerciales? (marque una)

- ○ Si
- ○ No

10. Según su experiencia, ¿qué recomendaciones adicionales puede dar para la implantación de código abierto o programas libres dentro de una organización?

# 4. RESULTADOS DE LA ENCUESTA

La encuesta fue realizada entre los meses de Junio 2005 y Julio de 2006 con un total de 191 encuestas. A pesar de ser un número bajo de organizaciones que fueron encuestadas, son suficientes para determinar una tendencia. Los resultados de cada pregunta se muestran a continuación:

## 4.1. ¿En qué tipo de organización trabaja?

Como se puede ver en la tabla I la mayor parte de los encuestado fueron empresas nacionales privadas, pero también aparecen datos de otros sectores.

**Tabla I. Resultados de la pregunta 1.**

| | |
|---|---|
| Empresa Privada Nacional | 105 |
| Empresa Transnacional | 31 |
| Empresa Multinacional | 13 |
| Organización Sin fin de lucro | 16 |
| Institución Educativa | 8 |
| Gobierno / Organización gubernamental | 16 |
| Otro (ONGs) | 2 |

## 4.2. ¿Conoce los términos Código Abierto y Programa Libre?

En la tabla II se puede ver que todos los gerentes o encargados de sistemas encuestados conocen sobre el código abierto, esto quiere decir que en Guatemala ya es conocido y que si no se utiliza, es por cualquier otra causa, pero no por desconocimiento.

**Tabla II. Resultados de la pregunta 2.**

| | |
|---|---|
| Si | 191 |
| No | 0 |

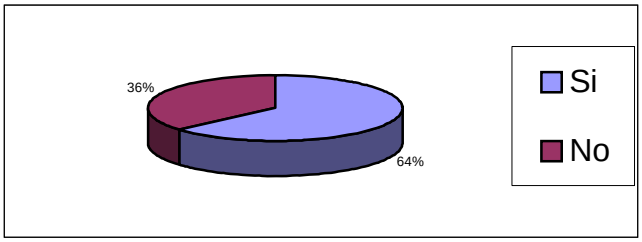
## 4.3. ¿En su organización utilizan programas o sistemas de código abierto o programas libres?

Para esta pregunta se puede ver tanto en la tabla III como en la figura 1 que a pesar que todos conocen sobre el código abierto, no todas las organizaciones lo utilizan, aunque el 64% de utilización es muy bueno en nuestro medio.

**Tabla III. Resultados de la pregunta 3.**

| | |
|---|---|
| Si | 172 |
| No | 69 |

**Figura 1. Resultado de la pregunta 3.**



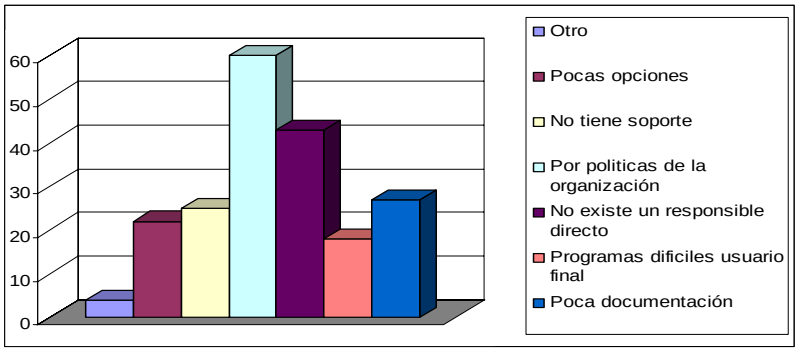

## 4.4. ¿Cuáles son los motivos principales por los que su organización no utiliza programas libres o de código abierto?

En esta pregunta solamente se tabularon los resultados de las opciones que fueron marcadas por lo menos una vez. Hay que recordar que esta pregunta solamente la respondieron quienes no utilizan programas de código abierto y que se pueden seleccionar varias opciones a la vez. Como se puede ver en la tabla IV las opciones están muy parejas, pero la que tiene más valor es la que nos indica que las políticas de la organización son las que más influyen en esta decisión. Esto puede verse con más claridad en la figura 2.

**Tabla IV. Resultado de la pregunta 4.**

| | |
|---|---|
| Otro | 4 |
| Pocas opciones | 22 |
| No tiene soporte | 25 |
| Por políticas de la organización | 60 |
| No existe un responsable directo | 43 |
| Programas difíciles usuario final | 18 |
| Poca documentación | 27 |

**Figura 2. Resultados de la pregunta 4.**



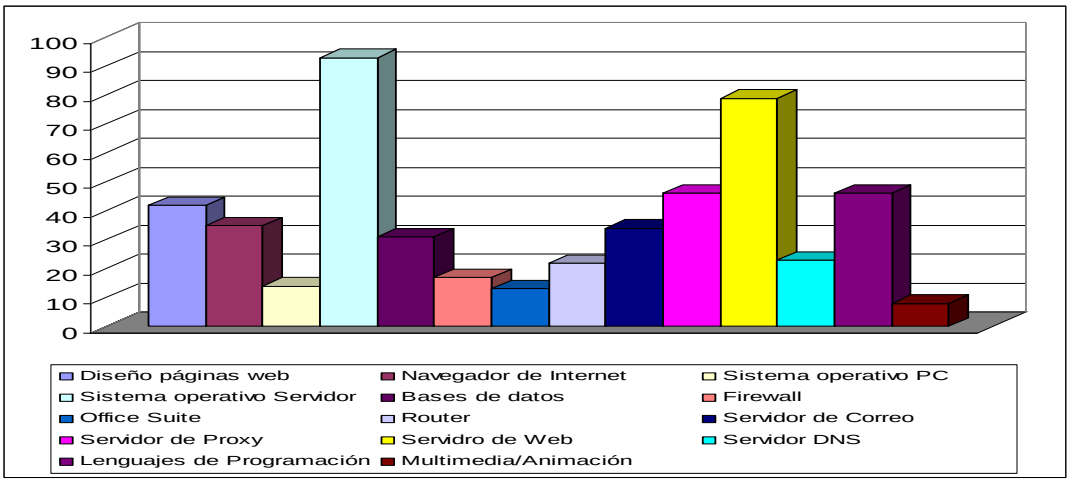

## 4.5. ¿En cuáles áreas se utilizan estos programas?

Para esta pregunta se puede ver en la tabla V y en la figura 3 que los usos más comunes son sistema operativo para servidor y servidor de web.

**Tabla V. Resultado de la pregunta 5.**

| | |
|---|---|
| Diseño páginas web | 42 |
| Navegador de Internet | 35 |
| Sistema operativo PC | 14 |
| Sistema operativo Servidor | 93 |
| Bases de datos | 31 |
| Firewall | 17 |
| Office Suite | 13 |
| Router | 22 |
| Servidor de Correo | 34 |
| Servidor de Proxy | 46 |
| Servidor de Web | 79 |
| Servidor DNS | 23 |
| Lenguajes de Programación | 46 |
| Multimedia/Animación | 8 |

**Figura 3. Resultados de la pregunta 5.**



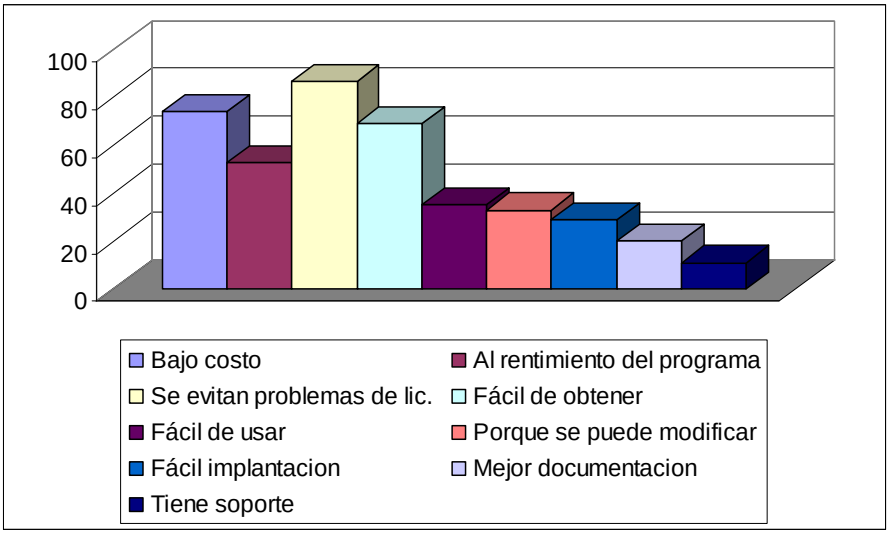

## 4.6. ¿Cuáles son los motivos principales por los que su organización utiliza programas libres o de código abierto?

En la tabla VI y la figura 4 se puede ver que evitar los problemas de licencias, el bajo costo y la forma tan fácil de obtener los programas, hacen que las organizaciones guatemaltecas utilicen el código abierto.

**Tabla VI. Resultados de la pregunta 6.**

| | |
|---|---|
| Bajo costo | 74 |
| Rendimiento del programa | 53 |
| Se evitan problemas de lic. | 87 |
| Fácil de obtener | 69 |
| Fácil de usar | 35 |
| Porque se puede modificar | 33 |
| Fácil implantación | 29 |
| Mejor documentación | 20 |
| Tiene soporte | 11 |

**Figura 4. Resultados de la pregunta 6.**



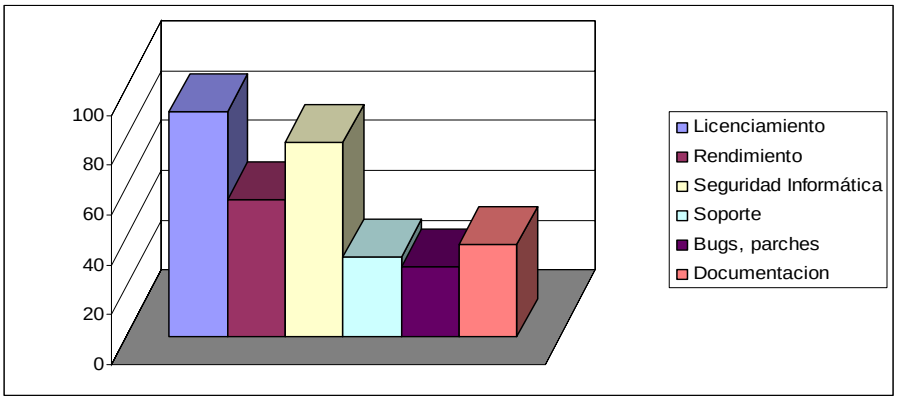

## 4.7. ¿A cuáles aspectos se les dio más importancia dentro de su organización cuando se decidió implantar un programa libre o de código abierto?

En la tabla VII y en la figura 5 se pueden ver las preocupaciones mayores de los gerentes al decidir si utilizan código abierto o no, y la opción más marcada es el licenciamiento.

**Tabla VII. Resultados de la pregunta 7.**

| | |
|---|---|
| Licenciamiento | 90 |
| Rendimiento | 55 |
| Seguridad Informática | 78 |
| Soporte | 32 |
| Bugs, parches | 28 |
| documentación | 37 |

**Figura 5. Resultados de la pregunta 7.**



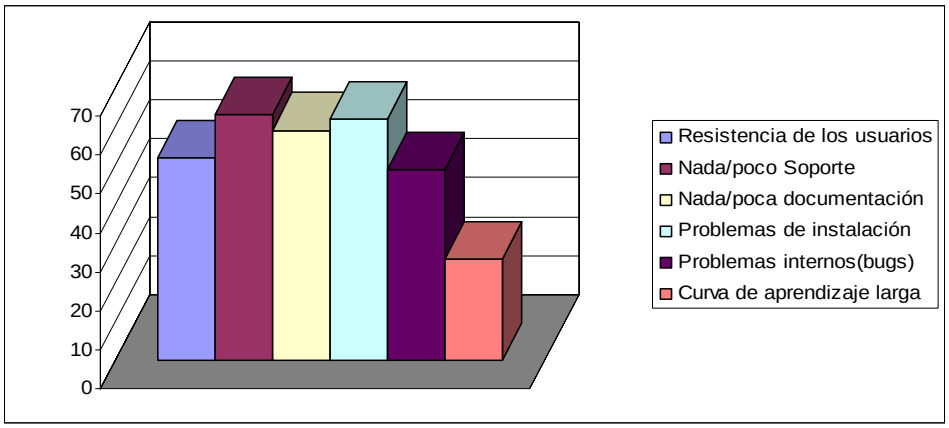

## 4.8. ¿Qué problemas encontró al implantar un nuevo programa libre o programa de código abierto?

Para esta pregunta se puede ver que las respuestas están muy parejas, por lo que podemos decir que casi todos los encuestados han tenidos los mismos problemas. Esto se puede ver con más claridad en la tabla VIII y la figura 6.

**Tabla VIII. Resultados de la pregunta 8.**

| | |
|---|---|
| Resistencia de los usuarios | 52 |
| Nada/poco Soporte | 63 |
| Nada/poca documentación | 59 |
| Problemas de instalación | 62 |
| Problemas internos(bugs) | 49 |
| Curva de aprendizaje larga | 26 |

**Figura 6. Resultados de la pregunta 8.**



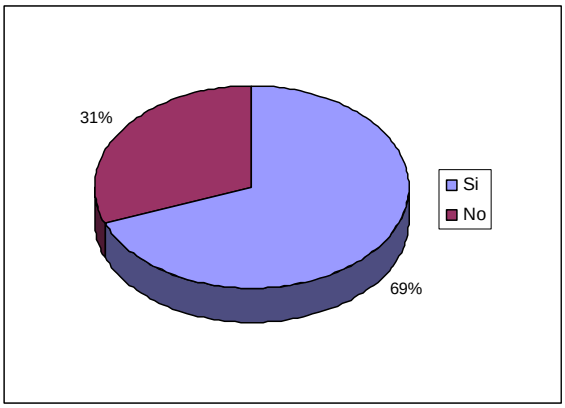

## 4.9. ¿Sabía Ud. que algunas compañías como IBM y Oracle han incorporado productos de código abierto dentro de sus productos comerciales?

Esta pregunta tenía como objetivo llegar a conocer que tanto los gerentes de informática sabían sobre lo fuerte que ha llegado a ser el código abierto, y como se puede ver en la tabla IX y en la figura 7 muchos no saben que lo que algunas grandes compañías hacen.

**Tabla IX. Resultados de la pregunta 9.**

| | |
|---|---|
| Si | 65 |
| No | 29 |

**Figura 7. Resultados de la pregunta 9.**




### 4.10. Según su experiencia, ¿qué recomendaciones adicionales puede dar para la implantación de código abierto o programas libres dentro de una organización?

En esta pregunta muchos volvieron a sugerir los puntos que estaban listados en la pregunta 8, la cual preguntaba en los problemas que habían encontrado los gerentes de informática al implantar un programa de código abierto.

Las recomendaciones más comunes fueron adquirir soporte, buscar bastante documentación y realizar pruebas del código abierto.

# 5. METODOLOGÍA DE EVALUACIÓN DEL USO DE PROGRAMAS DE CÓDIGO ABIERTO

En el presente capítulo se presenta una metodología que puede ser usada para evaluar que programa o sistema de código abierto debe implantarse en una organización, tomando en cuenta todas sus particularidades de este tipo de programas y las recomendaciones descritas en las encuestas. El objetivo principal es proveer una metodología cuantitativa que permita seleccionar el programa adecuado basándose en las necesidades de la organización y los objetivos de la implantación.

Para los programas de código abierto no existen en la mayoría de casos costos directamente asociados a licencias, soporte o actualizaciones, por lo que la evaluación de un programa o sistema no puede depender exclusivamente del calculo del Costo Total de Pertenencia, el cual es muy utilizado cuando se evalúan sistemas comerciales.

En algunos casos, sí existe el costo de algunos servicios como el de soporte, instalación o documentación, por lo que esta guía fue elaborada de tal forma que se puedan comparar los distintos programas, inclusive si ellos difieren en estas características.

Esta metodología consta de siete pasos los cuales son:

1. Definir el problema
2. Definir las funciones deseadas

3. Definir requerimientos
4. Ponderar las áreas a evaluar
5. Buscar los programas
6. Evaluar los programas
7. Calcular el punteo total por programa
8. Tomar la decisión

Es muy importante documentar todo el proceso, por lo que se recomienda tomar notas por escrito o bien, se puede crear una hoja electrónica de calculo que facilitará el ingreso de datos y los cálculos numéricos que se explicarán más adelante.

## 5.1. Definir el problema

El primer paso es definir el problema desde el punto de vista de la organización basándose en un incumplimiento específico de algún recurso físico o humano de la organización. Básicamente la definición del problema debe explicar la situación no deseada y dicha explicación debe ser concisa, objetiva y sin juicios. Este paso es necesario ya que nos ayuda a enfocarnos claramente en la búsqueda de un programa que resuelva el problema de la organización.

Un ejemplo de definición de problema es el siguiente: -- El gerente general de la organización ve con preocupación, que muchos trabajadores pierden el tiempo al navegar en Internet durante las horas de trabajo, por lo que desea un programa donde se pueda administrar y monitorear el acceso a Internet para poder saber quienes son las personas que pasan más tiempo en el Internet y poder llamarles la atención -- .

## 5.2. Definir las funciones deseadas

En este paso se deben definir las funciones específicas que deseamos que los programas a evaluar puedan ejecutar. Las funciones que cada programa puede realizar dependen del tipo de programa por lo que no se puede crear una lista genérica para utilizarla en todos los casos, se debe crear una lista basándose en la definición del problema del paso 1, con el objetivo de que las funciones puedan resolver los problemas allí planteados.

Al definir funciones debemos tomar en cuenta que algunos programas pueden no implementar una función específica, pero que a cambio pueden implementar varias funciones que puedan dar resultados equivalentes o no equivalentes pero aceptables que también puedan dar una solución al problema.

El listado de funciones se puede escribir en una hoja de papel o bien, se puede crear una hoja electrónica donde podemos listar las características en una columna escribiendo una por cada fila. Siguiendo el ejemplo usado en el paso 1, escribiremos en la tabla X, tres características deseadas para el programa que debe controlar el acceso a Internet, debido a que este es solamente un ejemplo no escriben todas las características deseadas.

**Tabla X. Ejemplo de hoja electrónica con funciones listadas.**

| **Funciones** | | | | |
|---|---|---|---|---|
| El programa debe ejecutarse como un servicio | | | | |
| Debe permitir la definición de usuarios | | | | |
| Debe permitir filtrar el acceso a Internet por usuario | | | | |

## 5.3. Definir requerimientos

Un requerimiento es una restricción que los programas deben cumplir para ser tomarlos en cuenta dentro del proceso de evaluación, normalmente las restricciones las imponen los equipos y sistemas existentes, pero también pueden existir restricciones que cumplan con la planificación a futuro del departamento de sistemas. Siguiendo con el ejemplo anterior un requerimiento puede ser: “El programa debe funcionar en el sistema operativo Windows 2003, ya que este es el que tenemos instalado actualmente en el servidor que va a ser utilizado para restringir el acceso”.

Se debe tener cuidado de hacer una lista corta y no confundir un requerimiento con una función deseada, si se falla en esta diferenciación se puede llegar a tener una lista muy larga de requerimientos que podrían llegar a discriminar muchos programas útiles en el proceso de evaluación. Si definimos correctamente una función y no la definimos como restricción, podemos todavía buscar dentro del un programa una o más funciones equivalentes, en cambio con las restricciones tendríamos que desechar el programa.

Usando el mismo ejemplo, en la hoja electrónica se deben agregar los requerimientos debajo de las funciones, un ejemplo de esto se puede observar en la tabla XI.

**Tabla XI. Ejemplo de hoja electrónica con los requerimientos.**

| **Funciones** | | | | |
|---|---|---|---|---|
| El programa debe ejecutarse como un servicio | | | | |
| Debe permitir la definición de usuarios | | | | |
| Debe permitir filtrar el acceso por usuario | | | | |
| **Requerimientos** | | | | |
| Corre en Windows 2003 | | | | |

## 5.4. Ponderar las áreas a evaluar

Un área es de un grupo de funciones de un programa que guardan algún tipo de interdependencia entre ellas, un ejemplo de un área es la “Seguridad” que agrupa todas las funciones que hacen seguro a un programa, por ejemplo, el manejo de usuarios y *passwords*, permisos, control de acceso, etc. En esta metodología se deben evaluar los programas usando las áreas que se listarán más adelante, pero primero se deben ponderar las áreas para especificar qué tan importante es para la organización dicha área.

Tomando en cuenta las características propias de los programas de código abierto, las áreas que se deben evaluar son: Seguridad, Rendimiento, Documentación interna, Documentación de usuario final, Acceso a información sobre el programa, Fácil instalación, Código fuente disponible, Tiene soporte técnico, El soporte técnico es de alta calidad, Existe soporte técnico local, Existen listas de correo para la distribución de información, Uso a nivel mundial, Planes de mejoras, Distribución de parches, Disponibilidad en español, Fácil utilización, Bajo impacto del cambio en los usuarios finales, Los manejadores del proyecto tienen renombre, El proyecto se mantiene activo y Existen cursos que puedan tomarse.

Para poder ejecutar este paso se debe tomar el listado de las áreas y se debe asignar un valor a cada una de ellas, el cual debe estar entre 1 y 5, donde 1 significa: muy poco, el 2 es: poco, el 3 es: normal, el 4 es: importante y el 5 es: muy importante.

Por ejemplo, para un banco que desea instalar un programa en una máquina con acceso a una red pública el área de “Seguridad” es muy importante y debe asignársele el valor 5, ya que todo programa para este tipo de organizaciones debe ser muy seguro; pero si vamos a instalar el programa en una red interna y el programa no va a realizar una tarea vital para la organización, el área de “Seguridad” puede no importar mucho, entonces se debe asignar el valor 1.

Para documentar este paso, en la hoja electrónica se deben agregar las áreas debajo de los requerimientos y en la columna de la derecha se debe escribir el valor que se le asigna a cada área.

Adicionalmente se debe realizar una sumatoria de la ponderación de todas las áreas, la cual debe ser colocada debajo de la columna de la ponderación. Esta sumatoria es necesaria ya que servirá en los siguientes pasos para hacer cálculos porcentuales sobre el total de las ponderaciones.

En la tabla XII se puede ver un ejemplo de cómo realizar este paso, con el listado pequeño de algunas de las áreas a evaluar, en la práctica se deben incluir todos las áreas para su evaluación.

**Tabla XII. Ejemplo de hoja electrónica con áreas y sus ponderaciones. El listado de áreas no esta completo por simplicidad.**

| **Funciones** | **Ponderación por Área** | | | |
|---|---|---|---|---|
| El programa debe ejecutarse como un servicio | | | | |
| Debe permitir la definición de usuarios | | | | |
| Debe permitir filtrar el acceso por usuario | | | | |
| **Requerimientos** | | | | |
| Corre en Windows 2003 | | | | |
| **Áreas** | | | | |
| Seguridad | 5 | | | |
| Rendimiento | 3 | | | |
| Documentación Interna | 4 | | | |
| .... | | | | |
| **Totales** | 12 | | | |

## 5.5. Buscar los programas

En este paso se debe iniciar la búsqueda de los programas que puedan ejecutar las funciones definidas en el paso 2 y también que cumplan con los requerimientos listados en el paso 3. Normalmente la descripción de los programas y la documentación que proporcionan son suficientes para determinar si cumplen con las funciones y los requerimientos. De no ser así, lo mejor es buscar más información en Internet y si esto no es suficiente, se deben realizar pruebas instalando los programas para verificar cada una de sus funciones y requerimientos.

Para documentar este paso debemos tomar la hoja electrónica donde se debe crear una columna a la derecha de la columna de ponderación de área, por cada programa encontrado y como título de la columna debemos poner su nombre. Luego se procede a realizar la evaluación de las funciones y requerimientos, escribiendo la palabra “Si” cuando el programa cumple con una

función o requerimiento y la palabra “No” si no lo cumple. Por el momento, solamente se evalúan los requerimientos y funciones.

Este paso se debe estar bien documentado, ya que en futuras ocasiones ya no es necesario volver a revisar las características de los programas que en esta ocasión se encontraron. Otra ventaja de documentar completamente este paso es que se podrá responder la pregunta sobre las razones por la que los programas fueron rechazados o aprobados por la evaluación.

La tabla XIII muestra un ejemplo con 3 programas agregados, donde uno de ellos no cumple con una función desea. Los nombres de los programas en esta tabla son ficticios.

**Tabla XIII. Ejemplo de hoja electrónica con tres programas agregados.**

| **Funciones** | **Ponderación por Área** | **Proxy 1** | **Proxy x** | **Proxy abc** |
|---|---|---|---|---|
| El programa debe ejecutarse como un servicio | | Si | Si | Si |
| Debe permitir la definición de usuarios | | Si | Si | No |
| Debe permitir filtrar el acceso por usuario | | Si | Si | No |
| **Requerimientos** | | | | |
| Corre en Windows 2003 | | Si | Si | Si |
| **Áreas** | | | | |
| Seguridad | 5 | | | |
| Rendimiento | 3 | | | |
| Documentación Interna | 4 | | | |

## 5.6. Evaluar los programas

En este paso se deben evaluar los programas y calificarlos en cada una de las áreas listadas en el paso 4, la calificación del programa se hace al asignarle un valor entre 0 y 5, donde 0 significa que el programa no provee

ninguna funcionalidad para el área calificada, 1 el programa es muy malo en esa área, 2 es malo, 3 es bueno, 4 es muy bueno y 5 es excelente.

A diferencia del paso anterior, para poder evaluar correctamente los programas no se debe consultar solamente la documentación, es necesario instalar el programa y realizar pruebas de tal forma se verifique cada función y requerimiento, esto permitirá una mejor asignación de la calificación para cada área.

Es posible asignarle a un programa una calificación mayor en un área que la ponderación que tiene dicha área, esto significa que el área puede ser ponderada muy baja porque no le interesa a la organización, pero al evaluar el programa obtuvo una calificación alta en dicha área. Opcionalmente la persona que evalúa podría tomar como valor máximo para la calificación el valor de ponderación del área.

Al final de la evaluación de un programa se debe calcular el total de calificación sumando todas las calificaciones que dicho programa obtuvo en todas las áreas.

Para documentar este paso, en la hoja electrónica se deben agregar las calificaciones de cada programa por cada área y la suma total. La tabla XIV es un ejemplo de cómo debe ingresarse la calificación de cada programa por cada área. También se puede ver que el tercer programa llamado “Proxy abc” no fue evaluado ya que no cumplió con dos funciones deseadas.

**Tabla XIV. Ejemplo de hoja electrónica con la calificación que obtuvo cada programa en cada área y la suma total por programa.**

| **Funciones** | **Ponderación por Área** | **Proxy 1** | **Proxy x** | **Proxy abc** |
|---|---|---|---|---|
| El programa debe ejecutarse como un servicio | | Si | Si | Si |
| Debe permitir la definición de usuarios | | Si | Si | No |
| Debe permitir filtrar el acceso por usuario | | Si | Si | No |
| **Requerimientos** | | | | |
| Corre en Windows 2003 | | Si | Si | Si |
| **Áreas** | | | | |
| Seguridad | 5 | **5** | **3** | |
| Rendimiento | 3 | **2** | **3** | |
| Documentación Interna | 4 | **4** | **5** | |
| .... | | | | |
| **Totales** | 12 | **11** | **11** | |

## 5.7. Calcular el punteo total por programa

En este paso se deben realizar dos cálculos, el primero consiste en una relación entre las ponderaciones de las áreas y el siguiente es el punteo de cada programa tomando en cuenta esa relación de las áreas. El significado de este calculo es que cada programa obtendrá una puntuación final que toma en cuenta la importancia de cada área para la organización, de tal forma que los programas que obtengan mejor punteo en la evaluación para las áreas más importantes, tendrán más oportunidad de ser seleccionados ya que tendrán punteos finales más altos. Otra forma de verlo es de que la relación del total es un factor de importancia, es decir, nos indica que tan importante es cada área para la organización.

El primer paso consiste en crear una columna extra que represente una relación que las áreas respecto del total de ponderaciones, a esta columna le llamaremos “Relación del total”. La formula para calcular esta relación es:

(ponderación del área) / (total de la suma de área). En la tabla XV se puede ver un ejemplo con la nueva columna llamada “Relación del total”.

**Tabla XV. Ejemplo de hoja electrónica la columna relación del total.**

| **Funciones** | **Ponderación por Área** | **Proxy 1** | **Proxy x** | **Proxy abc** | **Relación del total** |
|---|---|---|---|---|---|
| El programa debe ejecutarse como … | | Si | Si | Si | |
| Debe permitir la definición de usuarios | | Si | Si | No | |
| Debe permitir filtrar el acceso por … | | Si | Si | No | |
| **Requerimientos** | | | | | |
| Corre en Windows 2000 | | Si | Si | Si | |
| **Áreas** | | | | | |
| Seguridad | 5 | 5 | 3 | | **0.42** |
| Rendimiento | 3 | 2 | 3 | | **0.25** |
| Documentación Interna | 4 | 4 | 5 | | **0.33** |
| **Totales:** | 12 | 11 | 11 | | **1.0** |

En el ejemplo de la tabla XV se puede ver que la sumatoria de las características es 12 y esto representa el 100%, entonces una característica como la seguridad que tiene el valor 5, tiene una relación del 0.42 respecto al total ya que se divide 5 entre 12. Lo que significa este cálculo es de que la seguridad es tan importante para la organización que será parte de nuestra decisión final en un 42%.

El segundo paso consiste en agregar una columna por cada programa evaluado que servirá para asignar los puntos finales. La fórmula para esta columna es: (valor obtenido por el programa en un área) * (relación entre área y el total). El significado de este cálculo es de que cada programa obtendrá un punteo relativo a la importancia del área, entonces entre más alto sea el factor de importancia más puntos puede ganar el programa. Al terminar de calcular todos los punteos se debe hacer una sumatoria así obtenemos el total de calificación por programa

Por ejemplo en la tabla XVI se pueden ver dos columnas extras, una para cada programa, donde se agregaron los cálculos del punteo final que cada programa obtiene.

**Tabla XVI. Ejemplo de hoja electrónica con el punteo final por cada programa.**

| **Funciones** | **Ponderación por Área** | **Proxy 1** | **Proxy x** | **Proxy abc** | **Relación del total** | **Punteo Proxy 1** | **Punteo Proxy x** |
|---|---|---|---|---|---|---|---|
| El programa debe … | | Si | Si | Si | | | |
| Debe permitir la … | | Si | Si | No | | | |
| Debe permitir … | | Si | Si | No | | | |
| **Requerimientos** | | | | | | | |
| Corre en Windows… | | Si | Si | Si | | | |
| **Áreas** | | | | | | | |
| Seguridad | 5 | 5 | 3 | | 0.42 | **2.08** | **1.25** |
| Rendimiento | 3 | 2 | 3 | | 0.25 | **0.5** | **0.75** |
| Documentación Int… | 4 | 4 | 5 | | 0.33 | **1.33** | **1.37** |
| **Totales:** | 12 | 11 | 11 | | 1.0 | **3.92** | **3.67** |

Como podemos ver en la tabla XVI para el programa “Proxy 1”, la suma de los punteos normales era 11, pero al tomar en cuenta la relación que existe con la importancia de cada área, los valores cambiaron y las áreas más importantes para la organización obtuvieron mayores puntos finales que las áreas de menor importancia. Esto hizo que el punteo final del programa “Proxy 1” cambiara a 3.92.

En la tabla XVI también se puede ver que el programa “Proxy 1” obtuvo una buena calificación en el área de seguridad, mientras que obtuvo baja calificación en rendimiento; el “Proxy x” por otro lado obtuvo buenos resultados en rendimiento y mejores en documentación interna, pero al final el “Proxy 1” obtuvo más puntos porque obtuvo calificación alta en la seguridad, la cual es el área que más le importa a la organización.

## 5.8. Tomar la decisión

Si se realizan los cálculos tal y como se explicaron en el paso 7 lo único que queda por hacer es revisar la suma de las columnas de los punteos y el programa que tenga el número más grande es el programa ganador y el que debemos utilizar.

Como se puede ver esta metodología es objetiva ya que depende de los punteos y cálculos que se realizan, pero el proceso de evaluación debe de hacerse con mucho profesionalismo para poder establecer claramente los características deseada, las restricciones, ponderar las áreas y sobre todo al evaluar los punteos de cada programa.

Siguiendo con el ejemplo la decisión es utilizar el programa llamado “Proxy 1”, ya que obtuvo la mejor calificación total.

# CONCLUSIONES

1. Los programas publicados y certificados como código abierto tienen muchas particularidades, las cuales deben ser tomadas en cuenta en un proceso de evaluación de la utilización de dicho programa en cualquier organización.

2. Existen muchos tipos de licencias para los programas y antes de evaluar un programa para su uso, la licencia debe ser estudiada cuidadosamente para no encontrarnos con sorpresas después de que los programas ya están siendo utilizados.

3. Una de las mayores motivaciones de los creadores de programas de código abierto, es lograr crear programas de alta calidad usando un modelo de desarrollo colaborativo entre las personas que desean unirse libremente a cada proyecto.

4. Actualmente, el uso de código abierto en las organizaciones guatemaltecas esta muy difundido y entre las razones principales que motivan este uso están, el bajo costo de pertenencia, la reducción de problemas de licenciamiento y la fácil obtención de estos programas.

5. Las áreas de la informática donde se utilizan los programas de código abierto en Guatemala es muy amplia, pero existe una marcada tendencia al uso de estos programas en servidores.

6. Utilizar una metodología para la evaluación de programas de código abierto nos ayuda a organizar el proceso y a obtener resultados más objetivos que beneficiarán a la organización donde se utilizarán estos programas.

# RECOMENDACIONES

1. Las organizaciones guatemaltecas que desean iniciar un proceso de evaluación de uso de programas de código abierto, deben estudiar detenidamente sus necesidades para determinar si las particularidades de los programas de código abierto no imponen un obstáculo para implantar dichos programas.

2. A las organizaciones guatemaltecas de cualquier tipo, se recomienda iniciar procesos para evaluar programas de código abierto, ya que es una alternativa económicamente viable, ya que el costo de pertenencia de este tipo de programas es muy bajo.

3. Utilizar completamente la metodología del estudio de implantación de programas de código abierto, siguiendo todos sus pasos al pie de la letra, esto ayudará a obtener mejores resultados.

# BIBLIOGRAFÍA


1. Spinellis, Diomidis. Code reading: the open source perspective. Boston MA, United States. Adison Wesley, 2003. 598 páginas.

2. St. Laurent , Andrew M. Understanding Open Source and Free Software licensing. (Primera edición). O'Reilly Media, 2004. 224 páginas.

3. Golden, Bernard. Succeeding with Open Source. United States. Adison Wesley Professional, 10 de Agosto de 2004. 272 páginas.

4. Open Source Initiative Home Page. http://www.opensource.org . The open source definition version 1.9. Marzo 2006.

5. GNU's Not Linux Home Page. http://www.gnu.org. Marzo 2006.

6. FSF – The Free Software Foundations. https://www.fsf.org. Marzo 2006.