# Un tópico innovador en software: Ingeniería de software libre[1]

**Mauro Callejas Cuervo**[1]


## RESUMEN

Se pone a consideración un proceso que conlleva a la generación de *software* libre de alta calidad, a través de la conformación de *kernel* de trabajo discriminados en las diferentes etapas que componen el ciclo de vida de desarrollo de *software*; presenta, además, las funciones y principales artefactos que se deben obtener al aplicar este innovador paradigma (ingeniería de *software* libre). Por otro lado, se muestran algunas clases de *software* libre que existen en el medio informático, con el fin de poder concebir de manera diferente este revolucionario movimiento. Por último se comenta de manera breve el caso práctico de la aplicación desarrollada para la Arquidiócesis de Tunja, bajo algunos principios de este paradigma.


**Palabras claves**: Ingeniería de software libre, kernel, Herramientas de software libres, Metodologías.

## AN INNOVATIVE TOPIC IN *SOFTWARE*: FREE *SOFTWARE* ENGINEERING


## ABSTRACT

The purpose of this work is to consider a process to generate free *software* with a high Quality, through *Kernel's* work, which are discriminated throughout the different levels of the *software* development cycle life. Also, this work shows the functions and main devices, which should be used to apply this innovative paradigm (free *software* engineering). Additionally, it shows some types of free *software* used in the computing world, in order to understand the importance of this revolutionary trend. Finally, this work illustrates briefly a practical case with some principles of this paradigm developed at the Arquidiócesis of Tunja.


**Key words**: Free software engineering, kernel, tool of free software, methodology.

## 1. INTRODUCCIÓN

En la actualidad se está llevando a cabo una revolución informática basada en el tipo de licenciamiento del *software*, denominada *Software* Libre, que presenta métodos incipientes de programación colectiva y, además, no cuenta con un esquema de desarrollo que conlleve a generar productos de alta calidad, haciendo uso adecuado de las herramientas para tal fin. Este documento muestra un nuevo enfoque, que permite observar de manera diferente la forma de desarrollar *software* dentro de este movimiento, y, por ende, traza algunas pautas para que este trabajo de carácter colectivo se realice de manera eficiente. Así se pretende mostrar un nuevo juego lingüístico, como el concepto de *kernel* o núcleo de trabajo, sitio web para manejar el proyecto (la aplicación) y algunos términos más, que se describen en este documento.

La distribución del escrito está dada de forma tal que el lector tiene una guía práctica y secuencial para adentrarse en este revolucionario tópico de ingeniería de *software*. En primera instancia se presentan aspectos generales del trabajo enfocado bajo ciertas normas de ingeniería que se deben seguir; luego se plantean algunas funciones de los grupos que hacen parte del equipo de trabajo; posteriormente se describen de manera breve las clases de *software* libre y se muestran algunos ejemplos, y finalmente se dan algunas conclusiones y se plantea el trabajo futuro de investigación.

[1] Director del Grupo de Investigación en *Software* GIS, Investigador Principal Grupo *Software* Libre UPTC.

Artículo recibido en Septiembre de 2005
Aceptado en Octubre de 2005

[1]Artículo basado en la ponencia presentada en el «IEEE 3er Congreso Internacional en Innovación y Desarrollo Tecnológico» realizado en el la ciudad de Cuernavaca, Morelos, México, en septiembre de 2005.

Con este documento se pretende que el lector reflexione sobre el trabajo que aquí se resume, proponga sus puntos de vista y ponga a consideración sus tratados o postulados ante una crítica científica constructiva.

Algunos de los aportes aquí presentados son parte de la experiencia docente, en el desarrollo de asignaturas relacionadas con ingeniería de *software* y de proyectos de *software* libre dirigidos en trabajos de pregrado en ingeniería de sistemas [1], así como en la presentación de ponencias en eventos internacionales relacionados [2].

## 2. METODOLOGÍA Y MATERIALES

El diseño metodológico para la escritura de este artículo fue el siguiente: el tipo de estudio que se presenta, según Méndez [3], es de nivel exploratorio, ya que es el punto de partida para la formulación de otras investigaciones con mayor profundidad, que se describen en el aparte de trabajos futuros; el método de investigación es experimental, porque a través de la experiencia que se ha obtenido en el desarrollo de proyectos de *software* del género que atañe a este estudio se deducen ciertas realidades que deben ser puesta a consideración de la comunidad para poder ser replanteadas o fortalecidas; las fuentes y técnicas para la recolección de información son, en su mayoría, primarias (estudiantes y alguna parte de la comunidad que ha desarrollado proyectos de índole libre), y otras fuentes secundarias que se describen en las referencias bibliográficas; los resultados que aquí se presentan están dados en el desarrollo de cada uno de los capítulos que componen este artículo.

## 3. ENFOQUES O MÉTODOS DE DESARROLLO DE INGENIERÍA DE SOFTWARE LIBRE

La Ingeniería de *Software* Libre (ISL) permite que la metodología para el desarrollo de aplicaciones se lleve a cabo de manera amplia, ya sea utilizando un enfoque estructurado de análisis y diseño [4-6], un enfoque orientado por objetos [7] o algún otro tipo de paradigma; además no limita a los analistas y diseñadores a utilizar una técnica de modelado y diagramación, como UML [8] o el modelado estructurado, ni ofrece recomendaciones que permitan evaluar el nivel de calidad de una organización, como lo promueve The Capability Maturity Model, CMM [9]. Más bien se fundamenta en que se debe trabajar en equipo, con el fin de fomentar una mayor participación de elementos para el desarrollo óptimo de aplicaciones, sin dejar de lado la utilización de técnicas y herramientas que aquí se mencionan. Además, se debe tener en cuenta el tiempo y los recursos asignados para cumplir con las tareas involucradas, evitando la pérdida de tiempo o abandono de los proyectos.

Con la ISL se pretende promover el uso de sistemas operativos, lenguajes de programación, bases de datos y demás *software* de carácter libre para la creación de aplicaciones.

Otra de las partes que se analizarán en este documento es la posición de algunas personas que clasifican el *software* solamente como un producto, más no visualizan que detrás de él está la prestación de servicios como instalación, acomodamiento o reconfiguración a necesidades especiales y otras actividades que se mueven alrededor de esta nueva forma de ver el *software*.

### 3.1 Juego lingüístico sobre ingeniería de *software* libre

Un aspecto por tener en cuenta en el nuevo tópico de la ingeniería de *software* libre es el término *kernel*. ¿Por qué *kernel* y no simplemente grupo? La visión de *kernel* está dada en que es un grupo el que lo conforma, pero puede tener aportaciones valiosas a su alrededor, y allí es donde se evidencia el trabajo colaborativo o en comunidad, haciendo que cualquier aporte hecho fuera del grupo pueda ser compilado en su interior, con el fin de enriquecer el producto final. Lo anterior puede llegar a ser comparado con un sistema operativo (SO), donde el *kernel* administra y controla tanto *software* como hardware de manera rústica (modo consola), pero se puede y se debe llegar a crear herramientas que permitan al usuario una mayor interacción con el computador. Por ejemplo, programar un escritorio para Linux (KDE) hace que la interfaz sea más amigable al usuario y además ofrece un aporte significativo para mejorar la aplicación final.

El ciclo de vida para el desarrollo del *software* puede tomar algunos aspectos relacionados con metodologías ágiles o metodologías tradicionales [10], según la naturaleza del proyecto, pero haciendo un especial énfasis en que el desarrollo debe ser iterativo e incremental [11]. Tal y como se observa en la figura No. 1.

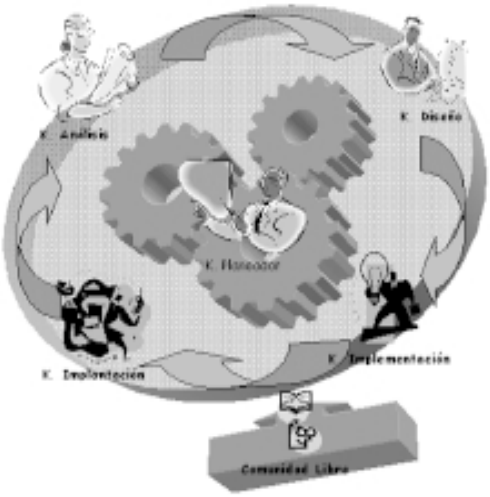


**Figura No. 1. Esquema de trabajo en Ingeniería de Software Libre. Se muestra un proceso cíclico, que evidencia como el kernel de planeación orienta a los demás kerneles en un perfecto engranaje y estos a su vez desarrollan sus propias actividades, dando su inicio a cada una de las iteraciones en el análisis y llegando a la implantación de cada uno de los módulos. Al finalizar cada iteración pasa nuevamente al análisis de otro de los módulos. Se observa además como la comunidad libre se encuentra fuera de la integración de los kerneles, ya que ésta es la que da sus aportes y punto de vista al grupo general de trabajo (aporta a cada uno de sus kerneles cuando sea el caso).**

También el término 'comunidad libre' juega un papel importante en este tópico; se refiere al grupo de personas que participan en actividades relacionadas con el análisis, diseño, implementación e implantación de aplicaciones informáticas, haciendo uso de herramientas de *software* libre, y quienes de manera voluntaria apoyan cualquier labor que tenga alguno de los *kerneles*. Esta comunidad debe conocer las políticas y los lineamientos elaborados por el *kernel* planeador para su participación en el proceso de desarrollo de la aplicación.

## 3.2 Aspectos relevantes en el desarrollo de *software* libre

Cuando se generan aplicaciones de *software* libre es importante tener en cuenta las herramientas que se utilizarán para la divulgación y manipulación del desarrollo del proyecto (sitio web), así como las partes que componen el equipo de trabajo, sus funciones específicas y algunos otros aspectos que tienen que ver con la ingeniería de *software* tradicional. En este aparte se hace una breve descripción de cada una de ellas.

### 3.2.1 Crear un sitio WEB

Para generar esta herramienta de visualización y manipulación del proyecto se deben seguir ciertos principios de usabilidad, que se describen muy puntualmente en Rodríguez de la Fuente [12]. En la web se plasmarán los avances, tareas pendientes, tareas asignadas, personas responsables y el cronograma de actividades fijado para el desarrollo total del proyecto; también debe mostrar la constitución de los diferentes grupos de trabajo colaborativo, con su líder correspondiente, quien será el responsable de la publicación de los diferentes hallazgos y productos resultantes del desarrollo de las actividades. El acceso de cada grupo está controlado para las diferentes actualizaciones, inserciones, modificaciones y otras actividades en el proceso de creación del producto[1]; el acceso para la comunidad libre participante será controlado y sus aportes serán recibidos en un espacio designado para tal fin.

### 3.2.2. Nombramiento de los diferentes grupos o *kernel*es de trabajo

Los *kerneles* que se integrarán tendrán funciones específicas y serán dirigidos por un líder nombrado por estos, quien hará las veces de vocero oficial de cada núcleo en las reuniones generales del proyecto. Los *kerneles* son:

***Kernel* de análisis**: nombrado *kernel* o núcleo porque está compuesto por una o varias personas que conforman la comunidad. Su líder guiará las diferentes actividades de análisis y será quien garantice la publicación de los productos generados en este grupo en el sitio web de la aplicación.

***Kernel* de diseño**: debe estar compuesto por un número limitado de miembros (dependiendo del volumen del proyecto); realizará las tareas de modelado lógico y físico de la aplicación y tendrá la responsabilidad de obtener los mejores modelos, poniéndolos a consideración de la comunidad libre, para captar sus aportes y recomendaciones.

[1]Por producto se entiende, el proyecto de software libre que se está desarrollando.

***Kernel* de implementación**: se divide en varios sub*kernel*es, organizados según su distribución geográfica, con el objeto de obtener mayor cooperación interna dentro de cada uno de ellos y así poder enviar sus aportaciones al grupo planeador, coadyuvando a que los productos que salgan de un lugar específico (ciudad, departamento, país, continente) sean de alta calidad y puedan ser publicados en la web y puestos a disposición de los demás equipos de trabajo.

***Kernel* de implantación**: compuesto por un grupo concreto de personas identificadas totalmente, pues será el encargado de poner a consideración el producto y capacitar al usuario final.

Todos estos grupos o *kerneles* serán coordinados por un grupo denominado ***Kernel* de Planeación**, responsable de dar el visto bueno para el arranque y finalización de las actividades de la comunidad de cooperación libre y de certificar la liberación al mundo del producto final.

#### 3.2.3. Asignación de funciones por *kernel*

A continuación se presentan algunas de las funciones más relevantes que se deben llevar a cabo en cada uno de los *kerneles*, así como también ciertos artefactos[2] que se crearán al ejecutar dichas funciones.

***Kernel* de planeación:**

- Mantener el sitio web.
- Coordinar las tareas de inicio y finalización de actividades.
- Elaborar el cronograma de actividades para el ciclo de desarrollo del producto de *software* libre.
- Asignar los controles de acceso al sitio web de la aplicación a los diferentes grupos de trabajo.
- Coordinar tareas de empalme entre los diferentes grupos de trabajo.
- Mantener el directorio de cada uno de los miembros y sus roles dentro de los diferentes grupos de trabajo, así como la bitácora de colaboraciones y participaciones en el desarrollo de la aplicación.
- Definir cuál es el eje geográfico del dominio de la aplicación.
- Llevar el control de versiones del producto.
- Crear las políticas para actualizar el producto.
- Documentar los procesos en los que está relacionado.
- Liberar el producto para su uso.

***Kernel* de análisis:**

- Contextualizar el domino de la aplicación.
- Recolectar información relevante para el proyecto, haciendo uso de las diferentes técnicas de levantamiento de información (entrevistas, encuestas, documentación histórica).
- Generar el documento de requerimientos del sistema.
- Crear los modelos o diagramas preliminares del análisis (diagramas de flujos de datos, diagramas de funciones, diagramas de actividades o en orientación a objetos casos de uso).
- Recolectar los aportes colocados en el sitio web de la aplicación, en cuanto a análisis del problema, con el fin de procesarlos y determinar cuáles se tendrán en cuenta para su implementación.
- Generar la documentación que se desprende de cada actividad de análisis.

***Kernel* de diseño:**

- Generar los modelos lógicos y físicos.
- Implementar el diseño inicial de la interfaz gráfica de usuario.
- Recolectar los aportes colocados en el sitio web de la aplicación, en cuanto a diseño, con el fin de procesarlos y determinar cuáles se tendrán en cuenta para su aplicación.
- Generar la documentación que se desprende de cada actividad del diseño.

Es importante aclarar que en la ISL las especificaciones de diseño no se toman por decisión unilateral del *kernel* o por convicción propia de alguno de sus miembros, sino de forma cooperativa, es decir, los demás *kerneles* y los miembros de la comunidad darán su aprobación en conjunto con el usuario final.

[2]Artefactos: son todos los elementos generados dentro del proceso de desarrollo de software (Documentos, diagramas, código, manuales y otros).

***Kernel* de implementación:**

- Crear los subgrupos de codificación, asignando políticas claras de desarrollo, es decir, asignación de codificación por módulo, por formularios u otros métodos de división del trabajo de programación de la aplicación.
- Generar estándares de codificación y construcción de las interfaces, es decir, plantillas que busquen la unificación de criterios de programación; para esto se sugiere la implementación y uso de patrones de diseño [13-14].
- Generar la codificación del producto.
- Desarrollar y llevar a cabo el plan de pruebas alfa.
- Generar la documentación que se desprende de la programación de los diferentes módulos de la aplicación. Esta documentación debe ser interna (dentro del código) y externa (en el manual del programador).

El trabajo que aquí se realiza es colaborativo, distribuido, y además debe haber un subgrupo que integre las diferentes partes de la aplicación ya programadas. Es de aclarar que un subgrupo puede estar trabajando en América y el otro en Europa o en cualquier parte del mundo. Una parte fundamental en el desarrollo de proyectos libres, es que todos los participantes dominen un idioma en común.

***Kernel* de implantación:**

- Instalar la aplicación desarrollada.
- Generar y llevar a cabo el plan de pruebas beta del producto.
- Capacitar a la comunidad que utilizará la aplicación.
- Documentar los procesos que se llevaron a cabo en esta etapa.

Si se analiza con detenimiento, en este enfoque innovador una de las principales funciones asignadas a cada grupo o *kernel* se fundamenta en la documentación exhaustiva de todos los procesos (análisis, diseño, implementación e implantación), pues es el valor agregado en un enfoque de desarrollo colaborativo y de libertad para generar este tipo de aplicaciones.

### 3.3. Clases de *software* libre

En el mundo del *software* libre se hace necesario identificar claramente los diferentes tipos de aplicaciones que existen. Por un lado están los programas ya desarrollados y que son herramientas de trabajo en el computador; auxilian las labores de los usuarios finales, como es el caso de OpenOffice (paquete de ofimática), que permite trabajar documentos de texto, hojas de cálculo, documentos HTML y presentador de diapositivas, entre otros; están también los navegadores, los antivirus, las herramientas para manipulación de archivos y otros más. Por otro lado están las herramientas para desarrollo de aplicaciones de *software* libre, que en resumidas cuentas se pueden clasificar en lenguajes de programación, servidores de aplicaciones y bases de datos. Y finalmente están los sistemas operativos. En seguida se presenta un listado de algunas de estas herramientas.

#### 3.3.1. Herramientas libres para el manejo del computador

Permiten la manipulación sencilla de los diferentes programas que están almacenados en el computador: escritorios, herramientas de ofimática, antivirus y navegadores, entre otros.

- **OpenOffice:** suite de ofimática libre desarrollada principalmente por SUN. Consta de un procesador de textos (Writer), una hoja de cálculo (Calc), un programa para presentaciones (Impress) y una aplicación para imágenes (Draw). Se puede encontrar más información sobre OpenOffice en http://www.openoffice.org. [15].
- **AbiWord**: procesador de textos libre. Está integrado en GNOME, aunque también existen versiones para otras plataformas, incluido Windows. Su página web se puede encontrar en http://www.abisource.com.
- **Mozilla**: proyecto iniciado por la compañía Netscape a finales de la década del noventa, tras liberar su navegador Netscape Navigator. Mozilla es una suite de internet que agrupa navegador, cliente de correo electrónico y compositor de páginas web. El proyecto Mozilla proporciona, además, un motor para

páginas Web Gecko y otra serie de herramientas muy populares, como ChatZilla. Más información en http://www.mozilla.org.

- **Evolution**: gestor de información personal para GNOME, desarrollado principalmente por Ximian. Se trata de una aplicación que reúne un potente cliente de correo electrónico, una agenda, un libro de contactos y un gestor de tareas. Información tomada de: http://ximian.com/products/evolution.

- **KDE**: acrónimo de K Desktop Environment (Entorno de Escritorio K). Entorno de escritorio completo, cuyo objetivo es acercar al usuario final a los sistemas GNU/Linux gracias a su amigabilidad y facilidad de manejo. Véase también GNOME. La página principal del proyecto KDE se puede encontrar en http://www.kde.org.

#### 3.3.2. Herramientas libres para el desarrollo de aplicaciones de *software* libre

**Lenguaje de programación:** conjunto de reglas semánticas y sintácticas utilizadas para dar instrucciones a un computador. Los lenguajes de programación permiten trabajar a un nivel de abstracción superior que con un código máquina, lo que facilita la creación y mantenimiento de programas informáticos. Existen cientos, si no miles, de lenguajes de programación. Algunos ejemplos son C, C++, Ada, Java, Pascal y COBOL [15].

- **Java:** moderna plataforma de programación creada por SUN en la década del noventa, que incluye un lenguaje de programación propio.

- **Jakarta:** subproyecto del proyecto Apache, cuyo objetivo es crear soluciones libres en Java, principalmente para el entorno web. Jakarta toma el nombre de la capital de la isla de Java, ya que el lenguaje de programación principal en el que está implementado es precisamente Java. Más información en http://jakarta.apache.org.

- **Ada:** lenguaje de programación diseñado con la seguridad en mente, principal razón por la cual fue encargado por el Departamento de Defensa de EE.UU. Es por eso que tiene una gran aceptación en la industria aeronáutica y aeroespacial. Su nombre proviene de Ada Lovelace, la primera *hacker* de la historia. El compilador más popular de Ada es GNAT [15].

- **Pascal:** lenguaje de programación de la década del setenta, escasamente utilizado hoy, aunque algunos de sus sucesores cuentan con amplio eco en la industria del *software*.

Otros lenguajes libres que se pueden mencionar son: **C, C++, Java, perl, tcl, python y PHP**.

**Bases de datos:** conjunto de datos que pertenecen al mismo contexto, almacenados sistemáticamente para su uso posterior. En este sentido, una biblioteca puede considerarse una base de datos, compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta; más información se encuentra en el sitio web: http://es.wikipedia.org/wiki/Base_de_datos.

- **MySQL:** esta base de datos robusta tiene soporte para distintos tipos de tablas, tales como ISAM, MyISAM, InnoDB y BDB (Berkeley Database). De estos, InnoDB es el tipo de tabla más importante, después del tipo predeterminado, MyISAM, y merece una atención especial. Las tablas del tipo InnoDB están estructuradas de forma distinta que MyISAM, ya que se almacenan en un solo archivo en lugar de tres, y sus principales características son la permisión de trabajar con transacciones y definir reglas de integridad referencial.

- **PostgreSQL:** base de datos que posee una trayectoria de casi dos décadas de desarrollo. Es el gestor de las bases de datos de código abierto más avanzadas en la actualidad; ofrece control de concurrencia multiversión, soporta casi toda la sintaxis SQL, incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario, y cuenta con un amplio conjunto de enlaces con lenguajes de programación como C, C++, Java, perl, tcl y python.

- **PointBase:** es una base de datos embebida 100% Java, especialmente indicada para aquellos proyectos que vayan a instalarse en dispositivos con pocos recursos, y es utilizada, en especial, para ejercicios académicos. ¿Qué más importante que aprender? Esta base de datos cuenta con una consola desde la cual es posible efectuar todas las operaciones ha-

bituales mediante la ejecución interactiva de sentencias SQL.

#### 3.3.3. Sistemas Operativos libres

Plataforma de *software* sobre la cual se alojan los demás programas de computadora y que permite manipular el *software* y hardware del computador.

- **FreeBSD:** es un sistema operativo libre para ordenadores personales, basado en CPU's de arquitectura Intel, incluyendo procesadores 386, 486 y Pentium (versiones SX y DX). También son soportados los procesadores compatibles con Intel como AMD y Cyrix. Actualmente también es posible utilizarlo en otras arquitecturas como Alpha/AXP, IA-64, PC-98, UltraSPARC y AMD64. Fuente de la información: http://es.wikipedia.org/wiki/FreeBSD.

- **GNU/Linux:** es la denominación defendida por Richard Stallman y otros para el sistema operativo que utiliza el *kernel* Linux en conjunto con las aplicaciones de sistema creadas por el proyecto GNU. Comúnmente este sistema operativo es denominado simplemente Linux. Tomado de: http://es.wikipedia.org/wiki/GNU/Linux.

En general, existen muchas otras opciones de trabajo con las diferentes clases de *software* libre, que ahora se encuentran en el mercado. Un sitio para visitar es: http://linuxshop.ru/linuxbegin/win-lin-soft-spanish/ index.shtml

## 4. CASO PRÁCTICO DE IMPLEMENTACIÓN DEL PROCESO DE ISL

La propuesta nace del trabajo realizado para varias empresas e instituciones colombianas, en las cuales se desarrollaron sistemas de información haciendo uso de software libre para su implementación, y de la necesidad de mejorar los procesos de ingeniería de software tradicional, que se mostraron cortos en el momento de su aplicación.

De las entidades en las que se ha implementado software, tratando de llevar a cabo la aplicación de la ISL, se pueden nombrar: Caracol Radio S.A., Acerías Paz del Río, Instituto Nacional de Salud, varias instituciones educativas (software para colegios e institutos de educación no formal) y Arquidiócesis de Tunja; comentaremos a continuación la experiencia de esta última.

En la Arquidiócesis de Tunja se implementó, en el marco de la Asignatura Trabajo de Campo II, de cuarto año de Ingeniería de Sistemas de la UPTC, la aplicación denominada «Sistema de control de información de la Arquidiócesis de Tunja», desarrollada con lenguaje PHP, base de datos MySql y utilizando como servidor de aplicaciones Apache, esta aplicación se encarga del manejo y control de la información de las diferentes parroquias que integran esta comunidad y de llevar el control de los diferentes eventos que allí se realizan. En el desarrollo de esta aplicación, los miembros del equipo se comunicaban a través del correo electrónico, adjuntando parte del código correspondiente al módulo asignado y poniendo a consideración del grupo su aportación; pero fue allí donde se notaron las falencias (limitación de espacio y tiempo), que fundamentaron la propuesta de crear equipos de trabajo (*kernel*es) y de condensar los avances en una página web que perteneciera al proyecto y en la cual se pudiera incluir los diferentes artefactos creados y se llevara un control riguroso del proceso, hasta obtener el producto terminado.

Los resultados obtenidos se reflejaron en la puesta en marcha de esta aplicación. La carta de la Arquidiócesis, de 'recibí a satisfacción', que reposa en las oficinas de la UPTC, demuestra el trabajo óptimo que se realizó. Queda demostrado que es necesario contar con un banco del proyecto en la web cuando los miembros involucrados en un equipo de desarrollo no se encuentran en el mismo sitio geográfico y que es importante la asignación de tareas claras para la consecución de los objetivos trazados. Además, la filosofía actual de desarrollo de aplicaciones en comunidad no posee una clara definición de tareas, responsabilidades ni límites de trabajo.

Con este caso práctico se refleja que el trabajo distribuido en *kernel*es es fundamental y que se debe apoyar en la creación de un portal para condensar los resultados que se vayan obteniendo, dando como resultado una forma

innovadora de plantear el desarrollo de aplicaciones de esta índole.

## 5. CONCLUSIONES

El trabajo con herramientas de *software* libre es hoy una tarea que está en crecimiento; se quiera o no, está ocupando un espacio muy importante en el mercado informático, razón por la cual la ingeniería de *software* debe y tiene que evolucionar hacia estas tendencias; en el documento aquí descrito se muestra este nuevo enfoque o fenómeno evolutivo.

Por otro lado, la programación libre está pasando por una etapa de cambio, como en aquellos tiempos en que se conocía el mundo de la informática como los laboratorios de programación de garaje; hoy se puede denotar que es un esfuerzo compartido y que ya se abandona la práctica tradicional de codificar aplicaciones y se nivela a un tópico más avanzado, como es la aplicación de principios básicos de ingeniería de *software*.

Se espera que la contribución hecha en este artículo motive el debate sano y cree la expectativa de investigar más a fondo las inquietudes que hayan surgido.

## 6. TRABAJOS FUTUROS

Dentro del grupo de investigación que viene trabajando se piensa plasmar una nueva metodología que reúna los pasos más importantes del ciclo de vida del *software*, bajo una panorámica libre en todo sentido; es así como se trabaja en la metodología MCAL (**M**etodología para la **C**onstrucción de **A**plicaciones **L**ibres), que se presentará en un futuro documento.

Otro de los trabajos para futuro es desarrollar una herramienta automática para el control concurrente de versiones (**CVS**: Concurrent Versions System) y de gestores de erratas.

## 7. REFERENCIAS BIBLIOGRAFÍA


[1] Callejas Cuervo M., Delgado Becerra J.G., El *software* libre como herramienta para el desarrollo de sistemas de información (Experiencia de una práctica empresarial en Caracol S.A.), Revista Ventana Informática, Edición Número 12, Enero-Junio de 2005, 15p.

[2] Callejas Cuervo M., et al., Ponencia El *software* libre como herramienta para el desarrollo de sistemas de información, IV Congreso Internacional de *Software* Libre GNU/LINUX, Universidad de Manizales, Manizales Colombia, 2005.

[3] Méndez C.E., Metodología: Diseño y desarrollo del proceso de investigación, Tercera Edición, McGrawHill, 2001.

[4] Witten J., Bentely L., Barlow V. M., Análisis y diseño de sistemas de información. Tercera edición, Editorial McGrawHill, 1996.

[5] Yourdon E., Análise Estruturada Moderna, Editora Campus, 1990.

[6] Kendall & Kendall, Análisis y Diseño de Sistemas, Tercera Edición, Editorial Pearson.

[7] Meyer B., Construcción de *Software* Orientado a Objetos, Prentice-Hall. 1998.

[8] Jacobson I., Booch G., Rumbaugh J., El Lenguaje Unificado de Modelado, Madrid, Editorial Addison Wesley, 1999

[9] M.C.Paulk, C.V.Weber, B.Curtis y M.B.Chrissis, The Capability Maturity Model: Guidelines for Improving the *Software* Process, Addison-Wesley, 1993.

[10] Pressman R., Ingeniería del *Software*. Un enfoque práctico, MacGraw-Hill, Quinta Edición, 2002.

[11] Larman C. Applying UML and Patterns. Perarson Edition. Prentice Hall. 2002. 590p

[12] Rodríguez de la Fuente, Pérez, Carretero y otros, Programación de aplicaciones web, Editorial Thomson, 2003.

[13] Gamma E., Helm R., Johnson R. y Vlissides J., Patrones de Diseño, Addison-Wesley, 2002.

[14] Design Patterns: Elements of Reusable Object Oriented *Software*. Addison Wesley, 1995.

[15] Matellan Olivera V., González Barahona J.M., Quirós de las Heras P., Robles Martinez G. Sobre *Software* Libre, Madrid, Editorial Dykinson S.L., 2004, 197p.



**Mauro Callejas Cuervo**

Ingeniero de Sistemas, Especialista en Ingeniería de *Software*, Tesista de Maestría en Ciencias Computacionales UNAB - ITESM. Universidad Pedagógica y Tecnológica de Colombia. Docente Investigador, Facultad de Ingeniería, Escuela de Sistemas y Computación, Director del Grupo de Investigación en *Software* GIS, Investigador Principal Grupo *Software* Libre UPTC. Tunja - Boyacá – Colombia, Avenida Central del Norte Km. 2 Vía a Paipa. Fax (098)7425268. maurocallejas@yahoo.com, mcallejas@tunja.uptc.edu.co.

REVISTA Ingeniería

Si desea publicar artículos técnicos en la **Revista Ingeniería** de la Facultad de Ingeniería de la Universidad Distrital Francisco José de Caldas. Puede utilizar como guía la plantilla para elaborar documentos técnicos en formato *Microsoft Word*. Esta se puede descargar como archivo de la página de la Facultad **http://ingeniería.udistrital.edu.co**

**Áreas de publicación**

- Comunicaciones y Televisión
- Telecomunicaciones –Teleinformática
- Instrumentación, Control y Automatización
- Electrónica de Potencia
- Dispositivos y Circuitos Electrónicos (FPGA, ASIC).
- Procesamiento de señales e imágenes
- Microelectrónica, Optoelectrónica y Fotónica
- Bioingeniería
- Tiempo Real
- Ingeniería de Software y Simulación
- Inteligencia Artificial, Sistemas Expertos y Robótica
- Gestión Informática
- Investigación de Operaciones
- Producción
- Logística
- Seguridad y Salud Ocupacional
- Geomática
- Catastro y Sistemas de Información Geográfica