# Índice general

# Capítulo 1

# Copyleft HW/SW Como Herramienta de Transferencia y Difusión

## 1.1. Introducción

Es indudable que el desarrollo tecnológico de un país se encuentra ligado al mejoramiento de la calidad de vida de sus habitantes, y que para que un país en vías de desarrollo se realice una transferencia tecnológica (y de conocimientos asociados a la tecnología que se transfiere) exitosa que permita desarrollar productos similares, pero ajustados al contexto socio-económico local, es necesario que el país cuente con la capacidad de absorber las habilidades, técnicas, información y organización asociadas a dicha tecnología. Esta absorción de conocimientos debe ser realizada por un gran número de personas para que la transferencia tenga un impacto significativo en la sociedad. La difusión de estos conocimientos es a nuestro modo de ver donde fallan los canales tradicionales de transferencia de tecnología y de conocimientos, ya que restringen el acceso a un grupo específico (representantes en el país de productos extranjeros, a quien pueda pagar por cursos de capacitación, o quienes puedan ingresar a centros de formación consolidados); para obtener la máxima difusión del conocimiento, es necesario considerarlo como un bien público (no rivalidad, sin posibilidad de exclusión), y como tal, el acceso a él, debe ser un derecho y por lo tanto, la sociedad debe garantizar los mecanismos de difusión para que llegue a los sectores de la sociedad interesados en él. De igual forma, es un deber, de los sectores que utilizan este bien común contribuir a su difusión, actualización, mejoramiento, y crecimiento.

En este estudio el conocimiento se refiere a las ideas intangibles, información y datos de todo tipo en el que el conocimiento es expresado u obtenido, como todo tipo de entendimiento adquirido a través de la experiencia o estudio ya sea propio, científico, académico o no académico. El conocimiento puede ser visto como una mercancía y como una fuerza constitutiva de la sociedad lo que evidencia la naturaleza compleja de este recurso; la adquisición y descubrimiento de conocimiento es un proceso social y personal. *El descubrimiento de conocimiento es un bien público, un tesoro que dejamos a futuras generaciones y el reto de nuestra generación es en mantener los caminos del descubrimiento abiertos*.

El acceso al conocimiento debe garantizarse sin restricción alguna a las personas interesadas, al considerar el conocimiento como un producto comercial se limita su difusión favoreciendo a un grupo de la población, lo que es inaceptable, ya que representa una forma de discriminación. En nuestro país, el acceso a la información ha ido aumentando gracias a la aparición de internet y a la gran cantidad de proyectos existentes que buscan difundir todo tipo de conocimientos. Sin embargo, para poder asimilar

este conocimiento se requieren ciertas habilidades que se adquieren después de un proceso de enseñanza. En Colombia el acceso a la educación superior de calidad es limitado, la cobertura de las universidades públicas está limitada por la inversión del estado. Por lo tanto, hacer del conocimiento un bien público requiere la intervención de diferentes sectores que garanticen y regulen el acceso, generen, clasifiquen y administren la forma en la que se expresa el conocimiento.

La diferencia entre nuestra propuesta y las existentes promovidas por los organismos gubernamentales, es el énfasis en la transferencia y difusión del conocimiento, del *saber hacer*, muchas políticas existentes se enfocan en la compra de equipo, lo que, como se vió antes no es el canal más eficiente para la transferencia tecnológica. Nuestros esfuerzos están enfocados a difundir un conocimiento básico en la concepción, diseño, implementación y diseño de sistemas embebidos; este conocimiento es el resultado de años de experimentación e investigación en el área; se realizó un trabajo previo de "adaptación" a las condiciones de la plataforma tecnológica local y al contexto económico y social con el fin de facilitar su absorción. Se espera que el libre acceso a su contenido aumente su difusión y llegue a todos los sectores de la sociedad que estén interesados en él. En la actualidad existen instituciones educativas que proporcionan conocimientos similares pero no tienen la profundidad del presente estudio y limitan el acceso a esta información cobrando altos precios para acceder a ella. De forma similar, el único centro de desarrollo tecnológico de la industria electrónica en Colombia el CIDEI, proporciona cursos de capacitación muy costosos (400 USD) sobre temas que ya han sido tratados en las universidades y no suministra información abierta a la sociedad (a pesar de que ha sido financiado con recursos de la nación por mas de 1 millón de USD).

La Universidad Nacional de Colombia, como el centro educativo más importante de Colombia tiene la obligación de cubrir este tipo de necesidades y hacer partícipes de los beneficios de su actividad académica e investigativa a los sectores sociales que conforman la nación Colombiana, por este motivo, todos los resultados del presente trabajo estarán a disposición de quien esté interesado sin restricción alguna.

En este capítulo estudiaremos las licencias *Creative Commons*, una forma de licenciamiento que permite asignar diferentes "permisos" de utilización a un determinado trabajo y pueden ser utilizadas en productos software y hardware. Exploraremos el movimiento de software libre y código abierto (FOSS) para identificar como contribuye a la transferencia tecnológica y como millones de usuarios a lo largo y ancho del mundo trabajan de forma conjunta para difundir, actualizar, mejorar, y aumentar las aplicaciones disponibles. Adicionalmente, se presenta una propuesta a la creación de un movimiento que funcione con políticas similares pero centrado en el desarrollo hardware.

## 1.2. El Conocimiento Como Bien Público

El análisis del conocimiento como bien común tiene sus bases en el estudio de recursos naturales compartidos (agua, bosques, fauna, flora). El recurso puede ser pequeño y ser utilizado por un grupo reducido (el televisor familiar), puede ser de nivel comunitario (campos de juego, bibliotecas, parques etc.) o pueden tener naturaleza internacional y mundial (océanos, atmósfera, internet, conocimiento científico, etc.), pueden ser delimitados (la librería comunitaria), transfronterizos (el río Amazonas, Internet, vida salvaje migratoria, etc.) o sin límites claros (el conocimiento, la capa de ozono, etc.). El conocimiento posee características que lo diferencian de los recursos naturales, mientras muchos bienes comunes ven comprometido su sostenimiento a medida que aumenta el número de usuarios, el objetivo de utilizar el conocimiento como bien público es lograr su máxima difusión [1].

El conocimiento, al igual que otros bienes públicos es utilizado de forma conjunta y es manejado por grupos de diversos intereses y tamaños. Para que este recurso sea útil se requiere una infra-estructura que permita su difusión, actualización, mejoramiento, realimentación y crecimiento. Pero ¿cómo administrar un recurso intangible y distribuido como el conocimiento? ¿Cómo coordinar acciones de miembros que se

encuentran localizados en diferentes puntos geográficos?, ¿Cómo coordinar acciones de miles de usuarios para garantizar un recurso útil y sostenible?, ¿Cómo convertir este conocimiento en motor de desarrollo y crecimiento empresarial?.

### 1.2.1. Auto-Gobierno

Ostrom [2] argumenta que la forma más eficiente de administrar un bien común es el auto-gobierno ejercido por beneficiarios de dicho recurso. Si estos beneficiarios son conscientes de la importancia de un uso eficiente y racionado que garantice la existencia de este bien y toda acción o decisión que se haga es formulada pensando en el beneficio común se crearán normas que garantizan el beneficio de todos los usuarios y la sostenibilidad del recurso. El auto-gobierno requiere acciones colectivas que "emergen cuando se requiere el esfuerzo de dos o más individuos para cumplir con una meta" [3]. Otro aspecto importante de las acciones colectivas es que son voluntarias por parte de los individuos [4]. El autogobierno requiere de la acción combinada de conocimiento y voluntad unido a normas institucionales consistentes. Ostrom [5] y Baland and Platteau [6], consideran que los atributos necesarios para que un recurso tenga una probabilidad alta de crear asociaciones de autogobierno son:

- Posibilidad de mejoramiento: El recurso no se encuentra en un punto tal que sea inútil crear una organización alrededor de él.
- Indicadores: Disponibilidad de indicadores confiables y válidos sobre la condición del recurso.
- Predictibilidad: El flujo de unidades de recurso es relativamente predecible.
- El recurso es lo suficientemente pequeño, teniendo en cuenta las tecnologías de comunicación utilizadas, para que los usuarios pueden tener un conocimiento preciso de los límites externos y de los micro-ambientes internos.

Y los atributos de los usuarios del recurso son:

- Prominencia: Los usuarios dependen del recurso para una parte importante de su sostenimiento.
- Entendimiento común: Conocimiento sobre el funcionamiento del sistema y como acciones individuales afectan a los demás y al sistema.
- Baja tasa de descuento: Para obtener un mayor beneficio a futuro se deben mantener tasas bajas de descuento del recurso.
- Confianza y reciprocidad: Los usuarios del recurso confían en que los demás mantendrán las promesas y se relacionarán entre ellos con reciprocidad.
- Autonomía: Los usuarios pueden determinar reglas sin la intervención de autoridades externas.
- Experiencia previa en organización y liderazgo local: Los usuarios poseen habilidades mínimas de organización y liderazgo a través de la participación en otras asociaciones locales u organizaciones en grupos cercanos.

De los anteriores atributos podemos deducir que para que pueda emerger un auto-gobierno asociado a un recurso debe existir un excelente canal de comunicación entre los usuarios de dicho recurso y que todos ellos deben conocer y aceptar una serie de normas (creadas por ellos mismos) que fueron formuladas pensando en el beneficio común. Un grupo de estas normas debe establecer mecanismos de resolución de

conflictos; esto es vital ya que toda la organización se basa en la confianza mutua y no puede permitirse que las relaciones entre los usuarios se deteriore. Adicionalmente, se requieren miembros que tengan claro que el éxito de sus acciones y el sostenimiento del recurso depende de sus acciones, las cuales deben realizarse teniendo en cuenta el beneficio común. El conocimiento no se ve afectado por la sustracción (uso) del mismo, todo lo contrario, el objetivo es obtener el máximo número de usuarios; sin embargo, este conocimiento puede llegar a ser obsoleto rápidamente, (en especial en el área de la electrónica digital) y por lo tanto inservible; para que esto no suceda, se requiere de su actualización para que refleje el estado actual en un área determinada.

**Ecuación Costo Beneficio de Ostrom**

La pregunta que trata de responder Ostrom en su estudio es ¿cómo emerge el auto-gobierno? [2] Para contestar esto formula una ecuación que describe el análisis costo-beneficio que un individuo realiza para participar en un gobierno comunal:

```
D < C (C1 + C2 + C3)
```

Donde $D$ es el incentivo al cambio y compara los beneficios de utilizar las reglas tradicionales (BO) frente a los beneficios de utilizar un nuevo grupo de reglas basadas en un gobierno comunal (BN)

```
D = BN − BO
```

$C1$ el costo anticipado asociado a la transición, $C2$ el costo de corto plazo para adoptar y apropiar las nuevas estrategias y $C3$ son los costos de largo plazo: mantenimiento, monitoreo y auto-gobierno. Para que se realice el cambio, el incentivo al cambio debe ser mayor que los costos asociados a el.

```
BN − BO > C
BN > BO + C
```

En la actualidad observamos que grandes multinacionales como Nokia, Motorola o Google, están participando y promoviendo el desarrollo de productos y aplicaciones [7] [8] [9] con la participación de la comunidad, lo que representa un giro de 180 grados en la políticas lideradas por multinacionales como Microsoft y Aple en las que todas sus creaciones están protegidas por licencias, acuerdos comerciales y contratos de exclusividad que restringen la participación en su desarrollo y aseguran el monopolio de los resultados de sus investigaciones. El movimiento de Software Libre [10] hizo posible este cambio, proporcionando herramientas de desarrollo y facilidades para que cualquier persona alrededor del mundo pudiera realizar sus propias aplicaciones, gracias a esto se han creado aplicaciones como el servidor web *Apache*, el explorador *Mozilla*, el sistema operativo *Linux*, aplicaciones como *gimp*, *openoffice*, librerías como *ncurses*, *Qt* y entornos de trabajo como *xfce*, *Gnome* y *KDE*. El trabajo realizado por miles de programadores para llegar a este estado ha sido enorme, y por esta razón los grandes multinacionales están poniendo sus ojos en estas iniciativas ya que les representa un gran ahorro de dinero, es decir, el costo de participar en estas iniciativas es mucho menor que el beneficio obtenido por participar en ellas.

*Linux Foundation* publicó recientemente un estudio donde calcula que el valor del kernel de Linux es de USD$1400 millones; y son necesarios USD$10.800 millones para desarrollar el stack completo de componentes desde cero; por este motivo, *Black Duck Software*[1] posee la más completa basa de datos de poryectos abiertos, representados en 200.000 proyectos, 4.9 billones de líneas de código, utilizando su detallado conocimiento de los proyectos abiertos y aplicando técnicas estándard de estimación de costos, estiman que el costo de desarrollo total del proyecto FOSS excede los USD$387000 millones y representa

[1] http://www.blackducksoftware.com líder mundial en el suministro de productos y servicios que aceleran el desarrollo software utilizando software libre

la inversión colectiva de mas de dos millones de desarrolladores al año. Un análisis adicional, el cual estima que el 10 % de las aplicaciones utilizadas para el desarrollo de aplicaciones IT, se pueden reemplazar por proyectos abiertos, lo que ahorraría mas de USD$22 billones al año.

De las experiencias obtenidas en la aplicación de la metodología de transferencia de conocimientos a la academia y el caso de estudio de transferencia a la empresa de base tecnológica emQbit, se estima que el tiempo necesario para que una empresa adquiera la experiencia necesaria para diseñar y construir sus propias placas de circuito impreso es de alrededor de 18 meses, el tiempo necesario para ajustar las herramientas de desarrollo y generar aplicaciones con la ayuda del sistema operativo Linux en plataformas propias requiere 12 meses. Adicionalmente, son necesarios otros 12 meses de experimentación para desarrollar habilidades y conocimientos en producción masiva. [2] Poporcionar diseños de referencia de placas de circuito impreso ayuda a reducir este tiempo de forma considerable, sin embargo, es necesario contar con habilidades que permitan entender el diseño, realizar los cambios necesarios y reproducir los pasos necesarios para construir la plataforma y escribir la aplicación.

Aunque muchos fabricantes de SoC comerciales proporcionan diseños de referencia incluyendo los archivos necesarios para realizar las modificaciones, es necesario contar con fuertes sumas de dinero para adquirir las licencias de las herramientas CAD utilizadas en su diseño y la complejidad de estas es tal (placas de circuito impreso de 6 o más capas, caminos de 0.1 mm y vías de 0.1 mm) que el prototipo de un diseño modificado costaría decenas de veces más que el valor comercial de la placa de referencia.

**Principios de Diseño**

El primer paso para gobernar un bien público es la identificación de los principios de diseño del recurso. Ostrom [1] después de dirigir un gran número de estudios empíricos sobre gobernancia de recursos de bienes comúnes, encontró una serie de principios de diseño que hacen que el recurso sea sostenible a lo largo del tiempo:

- Fronteras claramente definidas
- Las reglas deben reflajar necesidades y condiciones locales.
- Los individuos que están sujetos a estas reglas pueden modificar y participar en su elaboración.
- Las autoridades externas deben respetar el derecho de los miembros de la comunidad a crear sus propias reglas.
- Debe establecerse un sistema de seguimiento propio.
- Disponibilidad de un sistema gradual de sanciones.
- Acceso por parte de miembros de la comunidad a mecanismos de resolución de conflictos de bajo costo.
- Las actividades de apropiación, suministro, monitoreo y sanción deben ser organizadas en estructuras anidadas con múltiples capas de actividades

Es importante mencionar que no existe una combinación de principios que garantice el éxito, es necesario probar con este set para identificar los principios generales que deben incluir los sistemas robustos. Estos ocho factores fueron encontrados en la mayoría de las instituciones analizadas en cientos de

[2] Estas cifras fueron obtenidas de procesos reales donde un ingeniero con conocimientos básicos de manejo de herramientas CAD para fabricación de circuitos impresos y buen manejo de lenguajes de programación trabajó tiempo completo en estas tareas.

estudios y constituyen un buen punto de partida para comenzar esta investigación. Estos ocho puntos se encuentran fuertemente relacionados con los atributos que requiere un recurso y un usario de este recurso para que pueda emerger un auto-gobierno y buscan que todos los participantes tengan pleno conocimiento de las reglas que monitorean, regulan, fijan y controlan las actividades de los miembros.

**Movimiento FOSS**

Es interesante considerar como este pensamiento aplica a proyectos de código abierto. El software no es un recurso típico, porque no es sustraible, no existe un costo si un usuario decide usarlo o no; sin embargo, es un bien común, ya que es un recurso comunal que prospera o decae gracias a la contribución de sus miembros. Los participantes de este tipo de recurso son un grupo de contribuidores potenciales, en lugar del grupo de posibles propietarios. Un aporte de un usuario a mejorar el proyecto se traduce en un beneficio colectivo. Los contribuyentes pueden ser empresas que pagan a una persona para que adicione una nueva característica, sin embargo, ellos pueden decidir no compartir estos cambios para tener una ventaja competitiva. En la actualidad observamos las tendencias de multinacionales como Nokia o Motorola a participar, crear y promover proyectos de software libre, desde el punto de vista comercial resulta más rentable utilizar herramientas que han sido desarrolladas, probadas, y depuradas por miles de usuarios calificados que pagar a un grupo de decenas de desarrolladores para trabajar en sistemos propietarios.

En los proyectos de Código Abierto puede utilizarse una variación modificada de la ecuación de Ostrom [11]:

```
BC > BN + C
```

Donde $BC$ es el beneficio a contribuir, y debe superar el costo de contribuir $C$ más el beneficio de no contribuir $BN$. La comunidad del software libre ha creado un grupo de normas e instituciones muy sofisticadas alrededor de esta ecuación dando como resultado uno de los estructuras auto-gobernadas más exitosas. Los trabajos de la comunidad de software libre hacen que $C$ sea cada vez más pequeño, proporcionando:

- Herramientas de programación, depuración y librerías que facilitan el desarrollo de nuevas aplicaciones o mejoras a las ya existentes.
- Listas de discusión en donde los creadores de estas herramientas responden preguntas, y buscan en conjunto soluciones a problemas específicos
- Sistemas de control de versiones y protocolos de comunicación.
- Bases de datos de solución de problemas asociadas a las listas de discusión.
- Tutoriales, documentación, libros disponibles on-line.

Para contribuyentes no comerciales el costo de no contribuir es cero, pero, para contribuyentes comerciales $BN$ puede resultar muy grande, ya que pagar una nómina de programadores para que diseñen un sistema propio desde cero es mucho más costoso (en tiempo y en dinero) que pagar a pocos programadores pra utilizar y modificar proyectos ya existentes. La licencia más utilizada en los proyectos de software libre es la GPL, la cual, aunque no cubre todos los posibles tipos de uso, impone fuertes sanciones (escarnio público y consecuencias legales) cuando se utiliza y mejora un proyecto, pero no se comparten estos cambios con la comunidad.

## 1.3. FOSS

Como se mencionó anteriormente, el movimiento FOSS es la estructura auto-gobernada más exitosa y será tomada como referencia en el desarrollo de nuestra iniciativa de *hardware copyleft*. Richard Stallman [12], a mediados de los 80 inició un proyecto para desarrollar un sistema operativo abierto/libre basado en *unix* llamado *GNU* (GNU is Not unix); su principal innovación radica en un nuevo esquema de licencias unido a herramientas de colaboración basadas en Internet, lo que se convirtió en una nueva forma de bien público, donde los miembros de forma colectiva generan un bien público el *software*. El desafío de FOSS es realizar *acciones colectivas para crear y mantener este bien público*. A diferencia de la creencia popular, en este movimiento existen derechos de autor (*copyright*) y propiedad, sin embargo, algunos individuos involucrados en el proyecto, poseen derechos legales sobre el códico (recurso), tienen control sobre las nuevas versiones del software y pueden excluir a otros que aportan código a las nuevas distribuciones.

Los miembros del movimiento FOSS usualmente son programadores (y usuarios finales de software) que contribuyen (ya sea voluntariamente, o que se les pague para hacerlo) para producir un software bajo la licencia FOSS. La "situación de acción" que estos programadores encaran es, si en algún momento, vale la pena contribuir al desarrollo de este software. La interación de programadores trabajando de forma conjunta en Internet puede ser visto como un resultado que puede cambiar en el tiempo. Schweik y Semenov [13] identificaron tres estados de este tipo de bien una fase inicial; seguida por un estado de apertura y un estado más maduro de gran crecimiento (en términos de usuarios y participación); estabilización, donde el número de participantes (normalmente pequeño) no varía, o el protecto se estanca y muere (sin participantes). Para que un proyecto sea exitoso, no es necesaria la participación de un gran número de programadores, es frecuente encontrar grupos pequeños de programadores con un gran número de usuarios. La clave del éxito está en la disponibilidad de un programador para contribuir al esfuerzo colectivo de por lo menos un pequeño grupo de actores que producen y mantienen el software. En la Figura 1.1 se ilustra el marco definido por Ostrom y Hess [14] para analizar el conocimiento como bien común, aplicado al movimiento FOSS.

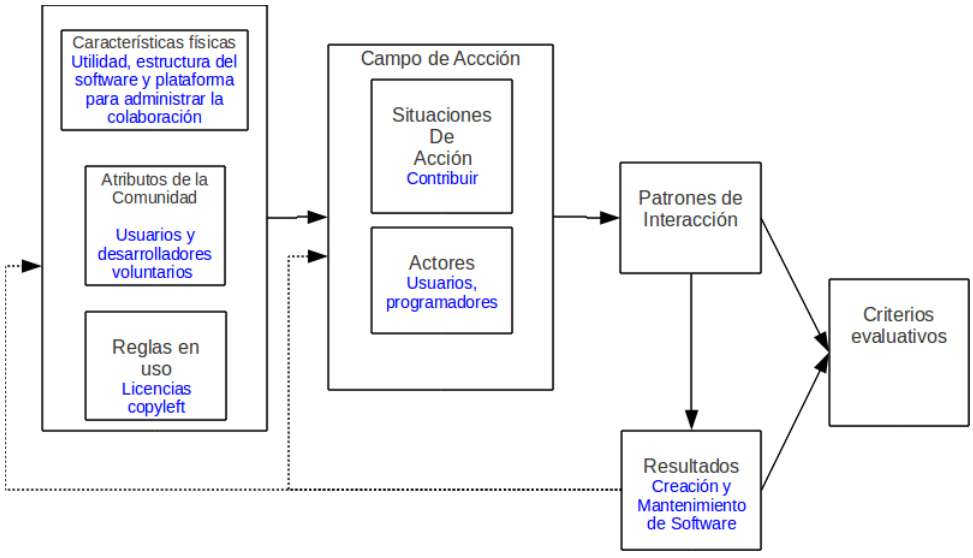


Figura 1.1: Marco para analizar el conocimiento como bien común aplicado al movimiento FOSS: Modificación de: [14] pág 46

**Reglas**

Stallman [15] sostiene que las propiedades digitales del software hacen posible considerarlo como un bien público, proporcionando a sus usuarios *la libertad* de utilizarlo, distribuirlo y modificarlo. Esta filosfía se basa en cuatro libertades que encarnan el principio *copyleft*:

- Libertad de ejecutar el software para cualquier propósito.
- Libertad de estudiar como funciona el programa, y cambiarlo para hacer lo que se desee. El acceso al código fuente es una condición para esto.
- Libertad para re-distribuir copias.
- Libertad para distribuir copias de versiones modificadas. Lo que permite que la comunidad se beneficie de estos cambios. El acceso al código fuente es una condición para esto.

Esta filosofía permite que los conocimientos y habilidades que el programador posee puedan ser transferidas a programadores, empresas, instituciones académicas y sociedades, ya sea en forma de producto o como herramienta de enseñanza. Esta actividad representa un proceso de transferencia tecnológica, en donde, el que suministra la tecnología proporciona todos los medios para que el receptor pueda absorberla y transformarla para satisfacer necesidades en su entorno social; sin embargo, para que la transferencia sea exitosa deben existir personas con la disposición y capacidad de asimilar los conocimientos asociados. Adicionalmente, Stallman ideó una forma de trabajar con las leyes de derechos de autor que proporciona una alternativa al licenciamiento tradicional. Las grandes multinacionales (como Microsoft o Apple) limitan el número de usuarios por licencia y hacen distribuciones de archivos binarios compilados en forma de ejecutables. El software con licencia *copyleft* estipula que cualquier modificación que se le realice adquiere los principios de licenciamiento del software original. La licencia *GPL*[3] (GNU General Public License) fue creada para implementar estos principios.

El proyecto de *código abierto* liderado por Bruce Perens y Eric Raymond, utiliza las propiedades de la licencia GPL adicionando un grupo de *derechos morales* dirigidos a conservar el trabajo del autor en su forma original, con el fín de conservar su autoría. Existe un gran número de variaciones de licenciamiento [4] para proyectos de software libre, lo que indica que al momento de licenciar un software el autor debe determinar que derechos retener y cuales ceder [16]. Las innovaciones del licenciamiento *copyleft* constituyen las *reglas en uso* que motivan a los programadores a participar en el movimiento FOSS, y establecen las bases de un régimen de propiedad pública.

La clave para mantener un bien público es la estructura de gobernanza de dicho recurso, el movimiento FOSS es un experimento en marcha, en dondes se prueba una mezcla imperfecta de liderazgo, mecanismos informales de coordinación, normas implícitas y explícitas junto con estructuras de gobernanza formal que evolucionan de tal forma que han logrado mantener unido este sistema tan camplejo [17]. Aunque no existe literatura que estudie las estructuras de gobernanza en el movimiento FOSS, algunas observaciones a los proyectos existentes en la actualidad indican que estas estructuras podrían incluir:

- Priorización de características a incluir en nuevas versiones del software.
- Definición de Reglas y procedimientos para evaluar y escoger nuevos aportes para que hagan parte de las nuevas versiones.
- Definición de Reglas y procedimientos para detectar y corregir errores en el software.

[3] http://www.gnu.org/licenses/licenses.html
[4] http://www.opensource.org/licenses/alphabetical

- Asignación y administración de tareas.
- Asistencia en la resolución de disputas entre miembros del equipo.

Estas reglas varían dependiendo del estado del proyecto, ya que, las actividades que se deben realizar en cada etapa son diferentes, en la etapa inicial, se establece la comunidad de programadores que definen las especificaciones del software, por lo que las tareas más importantes son el reclutamiento de programadores y la definción de una estructura flexible para el software. Cuando el proyecto alcanza una determinada madurez, lo importante es añadir funcionalidades y corregir errores.

**Atributos de la Comunidad**

La comunidad FOSS esta compuesta por usuarios y programadores voluntarios motivados por intereses tecnológicos, sociopolíticos, económicos y académicos. Desde el punto de vista tecnológico, resulta más efectivo en tiempo y en dinero, participar en un proyecto conjunto para la realización de una aplicación que satisface una determinada necesidad. La principal motivación sociopolítica es la creencia por parte del programador en la filosofía del movimiento, en la lucha contra el monopolio del software, adicionalmente, para que el proyecto permenezca durante mucho tiempo, es necesario que sea atractivo para vincular el mayor número de participantes y de esta forma sobrevivir en el futuro. Las motivaciones económicas y académicas pueden ser consideradas en forma conjunta ya que al participar en estos proyectos, los miembros adquieren o mejoran sus habilidades de programación ya sea revisando, leyendo y entendiendo el código existente o sometiendo a la revisión de los demás miembros sus contribuciones. Por otro lado, esta participación puede verse como una vitrina en donde el programador muestra sus capacidades y puede ser contactado por empresas para establecer relaciones comerciales. En los dos casos la participación es una forma de inversión y una forma de ganar experiencia y reputación [18].

En proyectos con un gran número de personas asociadas, solo un pequeño porcentaje de ellos realizan la mayor parte del trabajo. En la actualidad muchas empresas están participando de forma activa en estos proyectos, aportando programadores pagados para que contribuyan con el desarrollo y mejoramiento del software. Un estudio realizado a 25 firmas participantes en el proyecto FOSS *Linux Operating System* [19] indica que la tercera parte de los programadores encuestados son pagados por sus empleadores para participar, y que su motivación a participar es el interés propio en estandarización, disminución de costos, estrategias para debilitar a la competencia y esfuerzos para hacer sus productos compatibles con los productos derivados del movimiento FOSS.

**Atributos Físicos**

Existen tres categorías [16] en FOSS que pueden ser consideradas como atributos físicos:

1. La utilidad del software: Como vimos anteriormente, para que una persona participe en una actividad relacionada con el bien público, el costo por no hacerlo debe ser mayor que el beneficio por participar, es decir, el software debe representar una utilidad grande para que valga la pena participar en el proyecto.
2. El diseño o la estructura del software. Un código limpio (optimizado, documentado y óptimo) y modularizado permite el trabajo en paralelo, lo que reduce el tiempo de depuración y el requerido para implementar nuevas características.

3. La infraestructura que hace posible coordinar y administrar la colaboración y la producción: El uso de Internet para soportar herramientas de control de versiones (como *svn*, *cvs* y *git*) y la comunicación vía listas de discusión hace posible coordinar de forma eficiente los esfuerzos de cooperación; permitiendo:

   - Almacenar versiones de software.
   - La descarga de módulos.
   - La adición de nuevas contribuciones y la protección del código existente.
   - Generar un historial donde se documentan los cambios realizados a lo largo del tiempo.
   - Analizar funciones para identificar diferencias entre versiones.
   - Informar a los miembros del equipo cambios en los módulos del proyecto.

Esta infraestructura trabaja junto con las reglas-en-uso establecidas para proporcionar procesos que direccionen nuevos trabajos, un sistema para recibir y revisar contribuciones de nuevos módulos que pueden ser incluidos en futuras versiones. En conclusión, los proyectos FOSS evolucionan en el tiempo como resultado de la configuración de sus reglas-en-uso, atributos de la comunidad y atributos físicos relacionados con la estructura del software para hacer fácil la colaboración y herramientas efectivas para coordinación de equipos y manejo de contenido.

## 1.4. Hardware Copyleft

El concepto del *hardware copyleft* nació de un trabajo conjunto realizado con el proyecto *qi-hardware*[5] cuyo objetivo era crear un movimiento similar al FOSS pero aplicado al diseño hardware; en la actualidad, no existe un movimiento aceptado mundialmente que defina las propiedades que debe tener un proyecto hardware para ser considerado *abierto*; existen una serie de proyectos dispersos con diferentes características y grados de libertad [6] y cada uno de ellos interpreta a su manera lo que es un proyecto hardware abierto.

La iniciativa *Open Source Hardware* (OSH), es el busca aplicar principios generales de los ideales del hardware de código abierto con el fin de crear un estándar con el cual evaluar las licencias para diseños hardware. Esta iniciativa es apoyada por proyectos de hardware abiertos en su mayoría usuarios de la plataforma *Arduino* y por sitios encargados a hackear (aplicar ingeniería inversa) dispositivos comerciales, pero no existen muchos desarrolladores hardware que respalden esta iniciativa.

Nuestra propuesta presenta una visión diferente al proyecto iniciado con Qi-hardware, al incluir a la academia (considerada inútil por Qi-hardware) y buscar un grupo donde existan desarrolladores hardware y software (Qi-hardware solo tiene desarrolladores software) que contribuyan a la creación de plataformas físicas con su respectivo software de aplicación.

### 1.4.1. Licencias

Para manejar las licencias se trabajará con Creative Commons [7] organización no gubernamental sin ánimo de lucro que ha desarrollado una serie de licencias basadas en principios similares a los del movimiento de software libre y que pueden ser aplicadas a trabajos realizados en música, arte, video, texto y notas de clase. Para reproducir una plataforma hardware, es necesario suministrar los archivos de las

[5] http://en.qi-hardware.com/wiki/Qi:About
[6] http://en.wikipedia.org/wiki/List_of_open_source_hardware_projects
[7] http://creativecommons.org/

herramientas CAD de diseño y fabricación (PCB, Lista de materiales), por lo que esta iniciativa resulta adecuada para distribuir este tipo de proyectos. Estas licencias permiten que el autor de un trabajo conserve la propiedad intelectual (los derechos asociados a esta) pero posibilitando su copia y distribución, con la única condición de dar créditos al autor del trabajo original y que el trabajo derivado posea la misma licencia.

Se pueden elegir diferentes permisos dependiendo de los deseos del autor, para definir dichos permisos CreativeCommons proporciona una serie de preguntas que buscan determinar los derechos que se desean conservar y los que desean liberar. Las preguntas claves son:

1. ¿Los usuarios pueden copiar y distribuir su trabajo libremente?
2. ¿Es permitido a los usuarios crear trabajos derivados del contenido digital? Si es permitido, ¿Estas modificaciones deben tener la misma licencia que el trabajo original (esquema de licencia viral), o deben ser distribuidos bajo un esquema de licencias diferente?
3. ¿Es necesario atribuir el trabajo al autor original?

Estas preguntas son la base para definir un esquema de licencias modular. Existen cuatro bloques constructivos para las licencias Creative Commons estos son:

1. *Atribución* (**BY**) Se permite la distribución pero se debe dar crédito al autor.
2. *No comercial* (**NC**) Se permite la distribución del trabajo sin fines comerciales, si se desea utilizar este trabajo para obtener dinero es necesaria una autorización del autor.
3. *No a trabajos derivados* (**ND**) Permite la copia y la distribución del trabajo original sin modificaciones.
4. *Compartir de la misma forma* (**SA**) Exige que todo trabajo derivado del uso de proyectos con este esquema de licencias deben tener la misma licencia de los trabajos originales.

**Licencia CC - BY - SA**

La licencia *creative commons* **CC BY-SA** permite compartir, copiar, distribuir, ejecutar, comunicar públicamente la obra y hacer obras derivadas, con la condición de reconocer los créditos de la obra original de la forma indicada por el autor sin que esto implique que se cuenta con su apoyo o que apoya el uso que hace de su obra; si altera o transforma esta obra, o genera una obra derivada, debe distribuir la obra generada bajo la misma licencia. Este esquema de licencias ha sido elegida para licenciar las obras realizadas en este estudio.

Al elegir hacer el resultado de acceso libre, conservando los derechos de autor; se proporciona un bien público que puede ser utilizado por cualquier sector de la sociedad para suplir necesidades propias; su naturaleza viral asegura que se agragarán nuevos proyectos a este bien, la labor de la comunidad será definir que proyectos reúnen con los requerimientos de calidad para ser incluidos en los repositorios públicos.

**Licencia GPL**

Debido a que los proyectos hardware requieren la ejecución de un software que implemente la funcionalidad requerida; es necesario diferenciar los componentes de dichos sistemas, casi la totalidad de las aplicaciones software tomadas como referencia para adaptar las plataformas desarrolladas poseen la licencia GPL, por esta razón se conservarán las libertades de este tipo de licencias para el componente software.

- Libertad de ejecutar el software para cualquier propósito.
- Libertad de estudiar como funciona el programa, y cambiarlo para hacer lo que se desee.
- Libertad para re-distribuir copias.
- Libertad para distribuir copias de versiones modificadas; permitiendo que la comunidad se beneficie de estos cambios.

### 1.4.2. Características

Para que un proyecto sea considerado *hardware copyleft* es necesario proporcionar los archivos necesarios para cumplir con las libertades de las licencias GPL y CC-BY-SA. A nuestro modo de ver, la clave para garantizar dichas libertades está en suministrar la información necesaria para modificar los proyectos; para lo que es necesario poseer los conocimientos necesarios y tener acceso a los archivos de diseño.

Es posible reproducir un diseño realizado en cualquier herramienta CAD; sin embargo, existen dos factores que lo dificultan: el uso de herramientas propietarias y la complejidad del proyecto original. El uso de herramientas propietarias implica el pago de fuertes sumas de dinero para adquirir licencias y la complejidad del diseño puede incrementar de forma considerable el costo del prototipo. Debido al estado de la industria electrónica colombiana, es necesario reducir al máximo los costos para hacer atractiva la fabricación de productos propios[8]. Por lo tanto, es recomendable utilizar herramientas abiertas en el desarrollo de los proyectos *hardware copyleft*, en el capítulo **??** se presentó una metodología de diseño que solo emplea software abierto y será utilizada como referencia para el desarrollo de nuevos productos. Para reducir la complejidad del diseño se debe evitar el uso de componentes de dificil adquisición y utilizar 2 o 4 capas en la placa de circuito impreso. Los diseños de referencia que ya hacen parte del recurso utilizan otra restricción: no utilizar componentes con encapsulados BGA y QFN, ya que son difíciles de montar manualmente (esta restricción tiene sentido cuando se monta un prototipo o un número pequeño de unidades de forma manual) y es necesario utilizar técnicas de inspección con rayos X para verificar su correcto montaje.

#### Archivos necesarios para modificar un proyecto existente

Para modificar o reproducir un proyecto existente es necesario suministrar:

- Componente hardware
  - *Esquemáticos:* Diagramas eléctricos que representen el diseño completo.
  - *Netlist:* Archivo generado a partir del esquemático, contiene los elementos del circuito (señales y componentes), sus interconexiones, y asociaciones con representaciones físicas (dimensiones de líneas, footprints).
  - *Layout:* Representación física de la placa de circuito impreso, utiliza el netlist como entrada,
  - *Archivos de fabricación:* Archivos que son utilizados para la fabricación de las placas de circuito impreso, son útiles para os usuarios que solo quieran reproducir el proyecto.
  - *Archivos CAD* utilizados en la creación de formas especiales en las placas de circuito impreso, fabricación de la carcasa.
  - *Lista de componentes* Listado de los componentes utilizados en la placa de circuito impreso.

[8] Esto no implica que solo los proyectos más baratos hagan parte de la definición de un proyecto

- Componente software, código fuente de:
  - *Aplicación para grabar las memorias no volátiles*
  - *Bootloader*
  - *Kernel*
  - *Sistema de archivos*
  - *Aplicaciones*
- Archivos para dispositivos lógicos programables
  - *Aplicaciones*

## 1.5. Flujo de diseño utilizando *hardware copyleft*

Para determinar si es posible utilizar las plataformas de referencia creadas en este estudio en el desarrollo de nuevos dispositivos se utilizó la plataforma *ECB_AT91* [20] como punto de partida para la construcción de un monitor de signos vitales [21] y para la creación de un sistema de medición de la calidad del servicio eléctrico [22]. En el primer caso, un estudiante de posgrado agregó un módulo de comunicaciones para comunicarse con las tarjetas de adquisición de señales médicas y suministró puertos USB host para conectar módulos de comunicación WiFi, Bluetooth y 3G/GSM; demorando 6 meses en fabricar la nueva plataforma y en escribir la aplicación. En el segundo caso, la empresa emQbit, agregó una etapa de adquisición de señales análoga, un sistema de comunicación y un sistema de referencia geográfica basada en un GPS, empleando 6 meses en ello. Una reducción drástica frente al tiempo que demoró el desarrollo de la placa utilizada como referencia; esto es posible gracias a la posibilidad de utilizar los archivos de diseño de la plataforma, (los cuales, han pasado por un proceso de depuración y optimización) y el código fuente de las aplicaciones necesarias para la ejecución de aplicaciones Linux (Board Support Package BSP). lo que representa un ahorro de tiempo considerable ya que el proceso de concepción, diseño, puesta en funcionamiento y generación de aplicaciones para esta plataforma duró 18 meses. La figura 1.2 muestra una comparación del flujo de diseño cuando se utilizan o no plataformas *hardware copyeft*

Como puede apreciarse en la figura 1.2 la utilización de plataformas *hardware copyleft* afecta únicamente la etapa de implementación en el proceso de diseño (esto debido a que las primeras etapas no deben tener ninguna restricción tecnológica). Cuando se utilizan las plataformas *hardware copyleft* suministradas por este trabajo es posible reducir de forma considerable las tareas necesarias (cuadro azul de la figura) para la fabricación de un prototipo funcional; ya que al proporcionar los archivos necesarios para la modificación de las plataformas, se reduce de forma considerable el tiempo requerido para diseñar, construir, probar la placa de circuito impreso; adicionalmente, no es necesario dedicar tiempo para implementar las funciones básicas software y hardware ya que estas son proporcionadas.

El proceso de selección de la arquitectura se limita a elegir dentro de la gama de procesadores utilizados en este trabajo [9] el más indicado (económica y funcionalmente) para la aplicación, la tabla 1.1 resume las características de cada plataforma, indicando el costo de los componentes (costo de 50 unidades, a mayor cantidad menor el costo).

La opción más económica (según la tabla 1.1 es el uso de una plataforma basada en los SoC SAM7, los cuales incluyen en el mismo sustrato las memorias volátiles y no volátiles; la cantidad de memoria

[9] Es necesario aclarar que los procesadores utilizados en este trabajo son solo una muestra de la gran cantidad de referencias disponibles, por lo que es posible encontrar otros procesadores más económicos que implementen la funcionalidad deseada, las plataformas suministradas por este trabajo permiten ahorrar tiempo y dinero en la elaboración del prototipo y no necesariamente en el costo de fabricación en grandes volúmenes

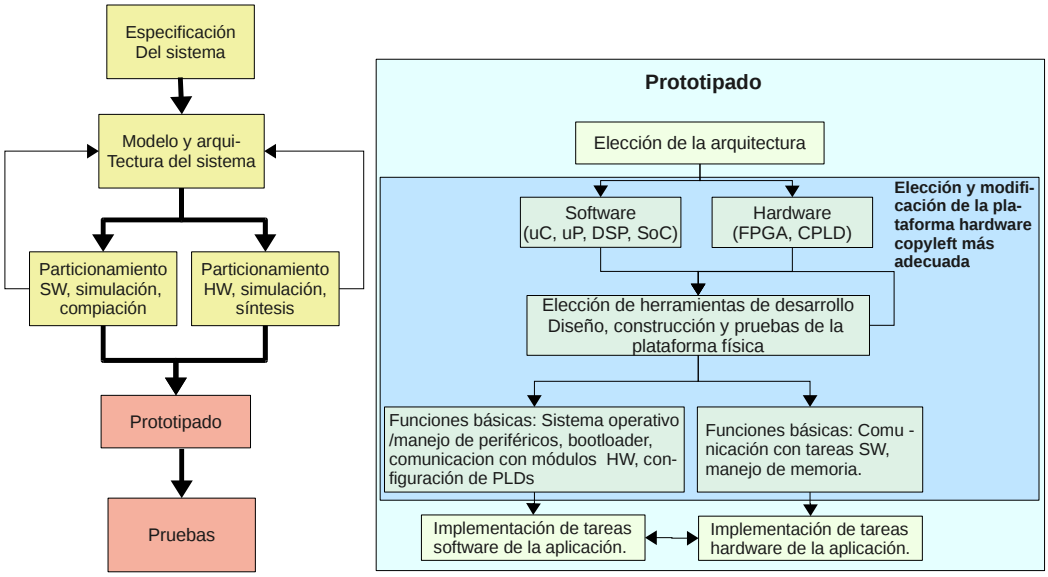


Figura 1.2: Flujo de diseño al utilizar hardware copyleft

Cuadro 1.1: Características de las plataformas de desarrollo *hardware copyleft* proporcionadas por el presente estudio

| **Plataforma** | **CPU MHz** | **CPU $USD** | **USB host** | **USB device** | **LCD** | **SPI/I2C/UART/** | **Mem Volátil $USD** | **Mem no Volátil $USD** | **FPGA $USD** | **Mínima config. $USD** | **Sistema Operativo** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 4.5 ECB_ARM7 | 33 | 3.5 | | X | | X | 0 - 6.5 | 0 | NA | 7.5 - 14.5 | RT - eCos |
| ECB_AT91_V1 | 180 | 18 | X | X | | X | 6 | 9 | NA | 33 | RT - Linux |
| ECB_AT91_V2 | 180 | 18 | X | X | | X | 6 | 9 | 20 | 53 | RT - Linux |
| ECBOT | 180 | 18 | X | X | | X | 6 | 9 | 10 | 43 | RT - Linux |
| ECB_BF532 | 400 | 17 | | | | X | 6 | 9 | 0 | 32 | uCLinux |
| SIE | 300 | 3 | | X | X | X | 6 | 12 | 10 - 23 | 31 - 44 | Linux |
| AndoroidStamp | 454 | 8 | X | X | X | X | 6 | 7 | NA | 21 | Linux |

disponible es de máximo 256 kBytes para la memoria no volátil y 64 kBytes de memoria RAM, lo que puede ser suficiente para muchas aplicaciones; sin embargo, las aplicaciones actuales que requieren capacidades multimedia, conexión a diferentes redes y manejo de sistemas de archivos no pueden implementarse en esta arquitectura.

El puerto de comunicaciones USB host permite la conexión de una gran variedad de periféricos (audio, comunicaciones, medios de almacenamiento) disponibles a bajos precios; reduciendo de forma considerable los costos finales. A manera de ejemplo, si una determinada aplicación necesita conexión inalámbrica a internet, una plataforma que no posea puerto USB-host necesita un controlador USB externo (cuyo costo puede ser de 10 USD) lo que aumenta la complejidad de la placa de circuito impreso y el costo de componentes.

Las FPGAs son muy costosas y elevan de forma considerable el consumo de potencia del dispositivo, por lo que deben ser utilizadas en aplicaciones donde sea estrictamente necesario, ya sea pora cumplir con restricciones temporales o para cumplir con funcionalidades especiales.

## 1.6. Comunidad linuxencaja

Uno de los objetivos del presente trabajo es crear una comunidad que utilice los frutos del presente trabajo, para adquirir conocimientos relacionados con el diseño de sistemas embebidos y crear nuevos productos que den solución a necesidades locales. Se creó el portal público *linuxencaja*[10] para reunir la documentación existente, proporcionar facilidades para la creación de nuevos proyectos, intercambiar opiniones con otros usuarios, almacenar los archivos generados por nuevos proyectos y mantenimiento de repositorios de código fuente y archivos de diseño de proyectos existentes. Este portal será el principal medio de comunicación y difusión, y el acceso a él no tiene restricciones de ningún tipo. A continuación se realiza una descripción de los recursos con los que cuenta la comunidad *linuxencaja*.

### 1.6.1. Recursos

#### Herramientas

- Lista de discusión en donde participan desarrolladores hardware/software, estudiantes y miembros de la comunidad para coordinar el desarrollo de nuevos proyectos y resolver dudas sobre conceptos relacionados con el diseño de sistemas embebidos.
- Base de datos de solución de problemas asociadas a las listas de discusión.
- Sistema de control de versiones *git* que permite el trabajo multi-usuario y la creación de repositorios para los proyectos.
- wiki que permite la elaboración de tutoriales, documentación sobre temas relacionados con el diseño de sistemas digitales o reportes del proceso de diseño de nuevos proyectos.

#### Diseños de referencia

Se cuenta con 7 tarjetas de desarrollo (ver descripción en el capítulo **??**) que cumplen con los requerimientos de los dispositivos *hardware copyleft* y se proporcionan los archivos necesarios para modificarlas;

[10] http://www.linuxencaja.com

estos diseños de referencia ya han sido construidos y probados en diferentes aplicaciones por lo que su funcionalidad es garantizada; todos ellos utilizan componentes que son fáciles de conseguir y no requieren grandes costos para su reproducción. En los repositorios del portal *linuxencaja* se encuentran las siguientes aplicaciones que pueden ser utilizadas para estudiar el funcionamiento de cada una de las plataformas:

- *loader y blink:* Aplicación que permite cargar binarios en la memoria interna del SoC (loader) y una aplicación que despliega un mensaje por el puerto serial de la plataforma y realiza un blink en un LED.
- *Drivers de dispositivos:* Ejemplos para aprender a escribir controladores sencillos para el sistema operativo linux.
- *Aplicaciones en espacio de usuario:* Ejemplos que permiten controlar los drivers desde espacio de usuario, y ejemplos de aplicaciones sencillas utilizando las facilidades del sistema operativo Linux.
- *Aplicaciones para configuración:* Algunas plataformas de desarrollo poseen micro-controladores de 8 bits o FPGAs que pueden ser programados o configurados desde la misma plataforma. Se suministra el código fuente de estas aplicaciones.

**Banco de proyectos académicos**

Cada semestre se realizan proyectos que utilizan alguna de las plataformas *hardware copyleft*, cada uno de ellos realiza la documentación completa del proceso de diseño y suministra los archivos necesarios para reproducir o continuar el proyecto. Este banco de proyectos representa un recurso donde se pueden encontrar soluciones a problemas comúnes de diseño y su realización ayuda a la formación de habilidades en diseño, escritura de informes técnicos.

**Documentación**

*Linuxencaja* proporciona documentación y tutoriales para:

- Artículos sobre la metodología de diseño y el programa académico generados por este trabajo.
- Proceso para proporcionar soporte a una plataforma embebida en el kernel de Linux.
- Comunicación entre tareas hardware (implementadas en PLDs) y software (implementadas en un SoC)
- Inicialización de SoCs.
- Proceso de fabricación y montaje (manual y automático) de placas de circuito impreso.
- Proceso de diseño que utiliza la metodología de diseño y el programa académico propuestos en este trabajo.
- Material para los cursos basados en la metodología propuesta (en construcción).
- Tutoriales sobre temas de interés sobre diseño software y hardware de sistemas digitales.

**Miembros**

*Linuxencaja* cuenta en la actualidad con diferentes tipos de usuarios: desarrolladores software, desarrolladores hardware, aficionados, estudiantes, industriales y administradores.

Los desarrolladores hardware y software se encargan de generar las plataformas de desarrollo y las aplicaciones software que las controlan e implementan su funcionalidad; el número actual de desarrolladores es pequeño, pero va en aumento gracias a los aportes de los estudiantes. Los aficionados, son personas a las que les gusta el diseño de sistemas embebidos, el desarrollo software, les gusta conocer el estado de la tecnologia, su papel es muy importante ya que por lo general, ellos se encargan de labores de mantenimiento, depuración y mejoramiento del material existente.

Los estudiantes, son miembros temporales que pueden o no continuar siendo parte de la comunidad después de cursar las asignaturas, sus aportes deben ser revisados con cuidado para verificar la calidad de su trabajo (muchos proyectos no se finalizan) y solo los mejores proyectos serán parte de los repositorios de la comunidad. A pesar de esto, durante los dos períodods académicos en los que se han vinculado estudiantes a la comunidad como parte de su proyecto de curso, se desarrollaron 5 aplicaciones de muy buena calidad y se reclutaron miembros activos, lo que muestra que esta política resulta muy beneficiosa para la comunidad.

Los industriales miembros de la comunidad se pueden dividir en dos grupos: los que hacen pequeños aportes en las listas de discusión, y no han creado proyectos nuevos, y los que participan activamente y aportan su trabajo a la comunidad. El primer grupo está conformado por empresas consolidades desde hace varios años (a excepción de la empresa emQbit) y su comportamiento es entendible debido a su naturaleza privada. El segundo grupo esta conformado por micro-empresas que han sido creadas por estudiantes que generaron nuevos productos con los recursos suminstrados por la comunidad y utilizan los resultados de los proyectos académicos.

Los administradores (2 personas) son personas que cuentan con conocimientos necesarios para configurar servidores y realizar mantenimiento a los servicios suministrados por el portal.

## 1.7. Conclusiones

El uso de plataformas *hardware copyleft* permite ahorrar tiempo y dinero en el desarrollo de aplicaciones proporcionando diseños de referencia modificables que pueden ser utilizados como base para futuros desarrollos.

La diferencia importante, entre el uso de plataformas *hardware copyleft* y el uso de plataformas de desarrollo disponibles comercialmente es la posibilidad de modificación, lo que permite optimizar los componentes a una aplicación determinada eliminando el costo de los componentes necesarios en la adaptación de estas plataformas genéricas; adicionalmente, se aumenta la oferta de productos generados localmente, lo que puede generar un aumento de la oferta de bienes y servicios en la manufactura de sistemas digitales, como ya ha ocurrido en otros países de la región.

Las plataformas suministradas como parte del recurso de la comunidad *linuxencaja* cubren un amplio rango de posibles aplicaciones y pueden ser utilizadas por cualquiera que este interesado para aumentar los conocimientos en el diseño e implementación de sistemas digitales, o para crear productos novedosos que satisfagan necesidades específicas de la sociedad. El ingreso de estudiantes a esta comunidad ha permitido el aumento del recurso, suministrando una nueva plataforma *hardware copyleft* y de proyectos de referencia que están en proceso de convertirse en productos comerciales.

La creación de este tipo de comunidades ayuda a difundir el conocimiento generado en la academia permitiendo el acceso a quienes estén interesados, lo que contrasta fuertemente con el modelo actual de

capacitación que requiere el pago de elevadas sumas de dinero para obtener este tipo de conocimientos. La incorporación de centros de formación e industriales a esta comunidad posibilita identificar necesidades de la industria y ofertas de la academia permitiendo la creación de un vínculo entre ambos sectores.

La creación de vínculos entre Universidad y empresa son muy difíciles de crear en nuestro país, en conversaciones realizadas con el prestigioso economista Clemente Forero ex-director de COLCIENCIAS, nos comentó de los programas que se realizaron en esat entidad para hacer que los industriales fueran a las universidades a buscar y como estos programas fracasaron debido a que las empresas y universidades nunca se pusieron de acuerdo, esto debido a que según Forero hablan idiomas diferentes. Durante la realización de este trabajo se pudieron establecer relaciones fuertes con 4 empresas tres de ellas creadas por egresados de la Universidad Nacional (emQbit, Santronics, Importex, em-electronics) y con dos empresas externas (SAR S.A y Microensamble S.A.), de estas relaciones se han generado pasantías, prestación de servicios a bajo costo y apoyo financiero para adquisición de equipo y manufactura de sistemas de desarrollo digitales; se intentó establecer vínculos con grandes empresas como CODENSA y TES AMERICA, pero este tipo de empresas no están interesadas en el desarrollo de productos sino en productos finales, este conocimiento es valioso ya que para poder establecer contacto con estas compañías es necesario tener desarrollos en una etapa que permita su producción masiva. De lo anterior podemos concluir que la mejor forma de crear vínculos entre la universidad y la empresa es haciendo que los estudiantes sean los actores principales de procesos de investigación e innovación orientados a la construcción de nuevas ideas de empresa (Spin Off Universitarias); sin embargo, el objetivo final no debe ser únicamente la creación de la empresa, sino brindar soporte para que esta pueda generar utilidades que puedan suministrar regalías a las universidades o la posibilidad de proporcionar bienes y servicios útiles en el ejercicio académico.

Nuestra experiencia con la empresa *emQbit* nos muestra que es posible inculcar en nuestros estudiantes la idea de generación de empresa, generar en ellos las habilidades necesarias para la producción de productos novedosos, y que con un acompañamiento adecuado por parte de la universidad (en el caso de emQbit como asesor externo) es posible que la empresa se mantenga actualizada y sobreviva en el tiempo, emQbit ha ayudado en el desarrollo de varios proyectos de investigación proporcionando materiales y equipos a costos reducidos (normalmente la empresas duplican sus precios cuando suministran productos a la Universidad Nacional), financiando parcialmente el desarrollo de plataformas *hardware copyleft*; adicionalmente ha vinculado a varios estudiantes de las universidades nacional y de los Andes en la modalidad de pasantía lo que les ha ayuda a formar personal para la propia empresa. Aunque este es el único caso que podemos presentar y no es posible generalizarlo, proporciona una referencia en este tipo de relaciones. La figura 1.3 muestra el modelo de transferencia utilizado con la empresa emQbit.

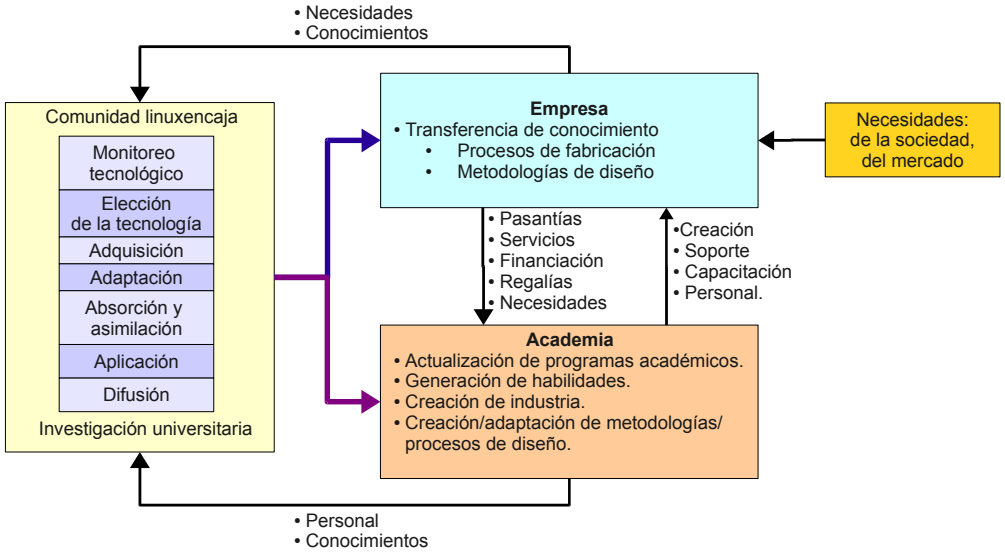


Figura 1.3: Modelo de transferencia tecnológica aplicado a la empresa emQbit

# Bibliografía


[1] Elinor Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action (Political Economy of Institutions and Decisions) (Paperback)*. Cambridge University Press, November 1990.

[2] Elinor Ostrom. Reformulating the Commons. *Swiss Political Science Review, Volume 6, Number 1*, 2000.

[3] Sandler Todd. Collective Action: Theory and Applications. *Ann Arbor: University of Michigan Press*, 1992.

[4] Ruth Meinzen-Dick, Monica Di Gregorio, and Nancy McCarthy. Methods for Studying Collective Action in Rural Development. *CAPRi Working Paper, no. 33. International Food Policy Research Institute, 2033 K Street, N.W., Washington, DC 20006. http://www.capri.cgiar.org/pdf/capriwp33.pd*, 2004.

[5] E. OSTROM. The Rudments of a theory of the Origins, Survival, and Performance of Common-Property Institutions. *BROMLEY, D.W. et al. (eds.) Making the Commons Work: Theory, Practice, and Policy. San Francisco, CA: ICS Press*, 1992.

[6] J .M. BALAND and J.P. PLATTEAU. Halting Degradation of Natural Resources. Is There a Role for Rural Communities? *Oxford: Clarendon Press*, 1996.

[7] Motorola. MOTODEV Project. URL: http://developer.motorola.com/platforms/mobile-linux/.

[8] Google. Android Project. URL: http://www.android.com/.

[9] Nokia. Maemo Project. URL: http://www.maemo.org/.

[10] Free Software Foundation. GNU General Public License. URL: http://www.gnu.org/copyleft/gpl.html.

[11] A. Cortesi. Elinor Ostrom, the commons problem and Open Source. *URL: http://corte.si/posts/opensource/ostrom/index.html*, 2009.

[12] Wikipedia. Richard Stallman. URL: http://es.wikipedia.org/wiki/Richard_Stallman.

[13] C. M Schweik and A. Semenov. The Institutional Design of 'Open Source' Programming: Implications for Addressing Complex Public Policy and Management Problems. URL: http://www.firstmonday.org/issues/issue8_1/schweik/, 2003.

[14] Charlotte Hess and Elinor Ostrom, editors. *Understanding Knowledge as a Commons: From Theory to Practice*. The MIT Press, December 2006.

[15] R. M. Stallman. *The GNU Operating System and the Free Software Movement Voices from the Open Source Revolution*. O'Reilly and Associates, 1999.

[16] Charles M. Schweik. *Understanding Knowledge as a Commons: From Theory to Practice*, chapter Free/Open-Source Software as a Framework for Establishing Commons in Science. The MIT Press, 2006.

[17] S. Weber. *The Success of Open Source*. Harvard University Press, 2004.

[18] Van Wendel, R. Joode, J. A. de Bruijn, and M. J. G. van Eeten. *Protecting the Virtual Commons: Self-Organizing Open Source and Free Software Communities and Innovative Intellectual Property Regimes.* Asser Press, 2003.

[19] R. A. Ghosh, G. Robles, and R. Glott. Free/Libre and Open Source Software: Survey and Study. Technical report University of Maastricht, The Netherlands: International Institute of Infonomics. URL: http://www.infonomics.nl/FLOSS/report/index.htm, 2002.

[20] C. Camargo. ECBOT y ECB_AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-diseño HW-SW. *VIII Jornadas de Computación Reconfigurable y Aplicaciones*, 2008.

[21] N. Rosas. Diseño e Implementación de un Sistema Embebido para la Adquisición y Transmisión de Señales Biomédicas a Través de la Red Celular. Master's thesis, 2011.

[22] E. Rodriguez, A. Calderón, D. Mendez, and C. Camargo. Sistema Integrado de Medición de Energía, Calidad del Servicio y Operación Remota. *MEMORIAS : VI Jornada de Distribución de Energia Eléctrica - ASOCODIS*, 2009.