Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

ANÁLISIS DE GRANDES ECOSISTEMAS DE SOFTWARE ABIERTO

TRABAJO FIN DE MÁSTER

Gabriel Orozco Martínez

2013

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MÁSTER

ANÁLISIS DE GRANDES ECOSISTEMAS DE SOFTWARE ABIERTO

Autor
Gabriel Orozco Martínez

Director
Juan Carlos Dueñas López

Departamento de Ingeniería de Sistemas Telemáticos

2013

Resumen

Las comunidades de código abierto han desarrollado con éxito una gran cantidad de software, aunque la mayoría de los usuarios prefieren utilizar software propietario. La evolución de las tecnologías de información han guiado a la comunidad de usuarios a observar y utilizar software de código abierto.

En este trabajo fin de master se define un ecosistema de software y los actores que se encuentra involucrados sobre una plataforma tecnológica que ofrece un número de soluciones de software o servicios. Con ciertas características que tienen la finalidad de analizar el conocimiento de un área y entender las tendencias que el impacto de la ingeniería de software está influyendo. En los modelos de negocio se identifican factores como: ciclo de vida del producto, área geográfica y ecosistemas diversos. Estos modelos de negocio están conectados con el resto de los actores y con el ecosistema como un todo de relaciones simbióticas, mientras que la plataforma tecnológica es estructurada de una forma que permite la participación y la contribución de los diferentes actores.

La Apache Foundation proporciona un ejemplo de cómo la complejidad se descompone en comunidades. Una de las primeras cosas que se pueden observar desde el sitio web de la comunidad Apache es que no hay una sola comunidad Apache. En su lugar, la página web Apache enumera varios proyectos que tienen un sitio web independiente y se comportan como una unidad relativamente independiente.

Un mecanismo para descomponer la complejidad en las comunidades de código abierto es a través de la creación de funciones especializadas. Las actividades que deban llevarse a cabo en las comunidades son diferentes. Ejemplos de ello son la creación real del nuevo código fuente, las pruebas del software en todo tipo de ajustes, encontrar, notificar y corregir errores; software de traducción a diferentes idiomas, escribir manuales y otros documentos, y creación y mantenimiento de las herramientas de apoyo, como un sitio web, CVS y la lista de correo. La mayoría de estas actividades están separadas unas de otras y actividades similares están agrupadas en roles, que a su vez están agrupados en proyectos o incluso en una sub-comunidad.

También se analiza y se describe a la Eclipse Foundation, la cual es una comunidad de personas y organizaciones que desean colaborar en el uso comercial de software open source. Sus proyectos se centran en la construcción de una plataforma de desarrollo abierta formada por frameworks extensibles, herramientas y tiempos de ejecución para la construcción, despliegue y gestión de software a través del ciclo de vida de un proyecto.

El propósito de Eclipse Foundation Inc., es avanzar en la creación, evolución, promoción y apoyo de la plataforma Eclipse y cultivar tanto una comunidad de código abierto y un ecosistema de productos complementarios, capacidades y servicios.

Por último, se presenta un análisis de eclipse y apache, que compara los dos enfoques, tanto por el soporte que ofrece la fundación, su infraestructura, los modelos de negocio, el ciclo de desarrollo, los roles y las licencias que aplican a sus proyectos, así como sus diferencias.

En conclusión con el análisis de estos grandes ecosistemas de código abierto, se ofrece una serie de recomendaciones para la creación de un ecosistema de código abierto.

Abstract

Open source communities have successfully developed a lot of software, but most users prefer to use proprietary software. The evolution of information technologies have led to the user community to observe and use open source software.

In order to master this work defines a software ecosystem and the actors who are involved on a technological platform that offers a number of software solutions and services. With certain features that are designed to analyze the knowledge of an area and understand trends that impact software engineering is influencing. In the business models identified factors such as product life cycle, geographic area and diverse ecosystems. These business models are connected with the rest of the actors and the ecosystem as a whole symbiotic relationship, while the technology platform is structured in a way that allows the participation and contribution of the different actors.

The Apache Foundation provides an example of how the complex is decomposed into communities. One of the first things that can be seen from the website of the Apache community is that there is a single Apache community. Instead, the Apache website lists several projects that have a separate website and behave as a relatively independent unit.

A mechanism to break down the complexity of open source communities is through the creation of specialized functions. The activities to be carried out in the communities are different. Examples are the actual creation of new source code, software testing in a variety of settings, find, report and correct errors ; software language translation, writing manuals and other documents, and creating and maintaining tools support, such as a website, CVS and mailing list. Most of these activities are separated from each other and similar activities are grouped into roles, which in turn are grouped into projects or even a sub-community.

It also analyzes and describes the Eclipse Foundation, which is a community of people and organizations that want to collaborate in the commercial use of open source software. Their projects focus on building an open development platform consists of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle of a project.

The purpose of Eclipse Foundation Inc., is to advance the creation, evolution, promotion and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities and services.

Finally, we present an analysis of eclipse and apache, which compares the two approaches, both for the support offered by the foundation, infrastructure, business models, the development cycle, roles and licenses that apply to their projects as well as their differences.

In conclusion the analysis of these large open source ecosystems, provides a number of recommendations for the creation of an ecosystem of open source.

Índice general

Índice de figuras

Siglas

Ecosistemas de Software (SECOs)

Global Software Development (GSD)

Software Production Line (SPL)

Software de código abierto (OSS)

Lineas de producción de software (SPL)

Open Source Initiative (OSI)

Licencia Pública General (GPL)

Licencia Pública General de Affero (AGPL)

Berkeley Software Distribution (BSD)

Licencia Pública de Mozilla (MPL)

Free Software Foundation (FSF)

Licencia de la Fundación Eclipse (EPL)

Licencia Pública Común (CPL)

Software como un servicio (SaaS)

PMC, comité de gestión de proyecto

Comités de Gestión de Proyectos (PMC)

Acuerdo de licencia Colaborador (CLA)

Repositorio de Subversion (SVN)

Convenio oficial de Donación de Software (SGA)

Propiedad Intelectual (IP)

Licencia Pública Eclipse (EPL)

Organización de Gestión Eclipse (EMO)

Acuerdo Committer Miembro (MCA)

Formulario de consentimiento Committer Empleador Eclipse (ECECF)

1 Introducción

Los ecosistemas de software son actualmente un término y una rama de investigación y desarrollo, con apenas unos años de su existencia ha heredado diversas características y tratamientos de otros tipos de ecosistemas como los digitales y de negocio.

Recientemente se ha sugerido que los Ecosistemas de Software (SECOs) son una forma efectiva de construir grandes sistemas de software sobre plataformas de software realizando la unión de varios componentes desarrollados por varios actores tanto internos como externos. En este entorno, la ingeniería de software se ha expandido fuera de sus límites tradicionales de compañías de software a grupos de compañías, personas privadas u otras entidades legales.

Esto difiere de las técnicas tradicionales para realizar outsourcing, en el que el actor principal no es necesariamente el dueño del software producido y contribuye con otros actores que no son los que contratan a los actores contribuyentes. Sin embargo, todos los actores coexisten de manera independiente, un ejemplo es el ecosistema de iOS en donde Apple provee de una plataforma para desarrollar, revisar y vender aplicaciones a cambio de una comisión de un 30% anual de las ganancias de las ventas de las aplicaciones. Esto es paralelo al ecosistema natural donde diferentes miembros del ecosistema (plantas, animales, o insectos) son parte de una cadena de comida donde la existencia de una especie depende de todas las demás.

También el Android de Google es un destacado ejemplo de un ecosistema de software. Dicho ecosistema ha ganado mucha importancia comercialmente: por ejemplo, en el 2012, se vendieron más teléfonos inteligentes que ordenadores personales.

Mientras los ecosistemas de software indiscutiblemente están ganando mucha importancia, su campo de investigación todavía se encuentra a una temprana edad, se comenzó en él 2005 con los estudios de Messerschmitt y Szyperski y ahora con una conferencia que se realiza todos los años, el International Workshop on Software Ecosystems IWSECO.

Los ecosistemas de software (SECO´s) es una disciplina emergente que estudia las relaciones entre las empresas de la industria del software. Las empresas trabajan en cooperación y competitividad con el fin de alcanzar sus objetivos estratégicos. Ellos deben participar en una nueva perspectiva, ya que también incluye la tercera parte de motivaciones y de los movimientos en el ecosistema, además de su propio punto de vista empresarial. Inspirado en las propiedades de los ecosistemas naturales y de

negocios, SECO cubre los aspectos técnicos y de negocio de desarrollo de software, así como de colaboración entre las empresas.

Estas relaciones se basan frecuentemente en una plataforma tecnológica común o de mercado y opera a través del intercambio de información, recursos y artefactos. El ecosistema móvil es una larga red que contiene: operadores móviles de red, proveedores y compradores de estos servicios.

Debido a las diferentes necesidades de los lados de la plataforma, la fijación de precios en un mercado es difícil y requiere de un modelo de negocio diferente. En el caso de las plataformas móviles, los usuarios y los proveedores de contenidos son los dos lados del mercado. La plataforma móvil no es el dispositivo móvil, es el mercado de aplicaciones móviles.

Multi-homing describe una situación en donde existen varias plataformas económicas que luego compiten en dos lados diferentes, y los dos lados son libres de escoger en cual plataforma desean desarrollar su aplicación, un ejemplo de esto un desarrollador puede realizar multi-homing en Windows o Apple o Android, a su vez un usuario es multi-homing cuando escoge un móvil y S.O que desea.

1.1 Objetivo

El objetivo principal de este trabajo es identificar los principios, mecanismos y operativa de la Apache Foundation y Eclipse que permite el establecimiento de ecosistemas de software autosostenibles basados en el código abierto.

2 Definición de Ecosistema de Software

El término ecosistema de software fue presentado por el Messerschmitt y Szyperski en 2003. Los autores comienzan a ver la industria del software como un ecosistema, la introducción de este software es la industria indispensable, y explican ecosistema de software como el software y su contexto, incluyendo los aspectos tecnológicos y no tecnológicos. Se requieren profesionales para hacer frente a desafíos en todas las etapas de estas cadenas, incluyendo análisis, diseño, implementación, aprovisionamiento y operación.

Un Ecosistema de Software se define como la interacción entre un grupo de actores que se encuentra sobre una plataforma tecnológica común que se torna en un número de soluciones de software o servicios. Cada actor es motivado por una serie de intereses o modelos de negocios conectados con el resto de los actores y con el ecosistema como un todo con relaciones simbióticas, mientras que la plataforma tecnológica es estructurada de una forma que permite la participación y la contribución de los diferentes actores.

En otras palabras, el Ecosistema de Software (ES) provee de posibilidades para que los actores se beneficien de su participación en el ecosistema. Los tipos de beneficios pueden variar dependiendo del actor y de la naturaleza del ecosistema.

Una definición formal de un ecosistema podría ser:

Manikas, Software Ecosystem, Technical Report no. 2012/02, ISSN: 0107-8283, paginas 10-11.

Messerschmitt y Szyperski (2005) es la definición más antigua de SECO en la literatura se encuentra en referencia al libro de SECO publicado en 2005.

"Tradicionalmente, un ecosistema de software se refiere a una colección de productos de software que tienen algún grado dado de relaciones simbióticas." (Messerschmitt y Szyperski, 2005).

Slinger Jansen (2009) define un ecosistema de software como:

"Un conjunto de empresas que funcionan como una unidad y que interactúan con un mercado común para el software y los servicios, así como las relaciones entre ellos. Estas relaciones se basan con frecuencia por una plataforma tecnológica común o de mercado y operan a través del intercambio de información, recursos y artefactos. " (Jansen et al., 2009b)

Y para **Jan Bosch (2009)**, ecosistema de software se define como:

“Un ecosistema de software consiste en el conjunto de soluciones de software que permiten, apoyar y automatizar las actividades y transacciones de los actores en el ecosistema social o de negocios asociados y las organizaciones que ofrecen estas soluciones.”(Bosch, 2009)

“Un ecosistema de software consiste en una plataforma de software, un conjunto de desarrolladores internos y externos y una comunidad de expertos en los sectores de servicio a una comunidad de usuarios que componen los elementos pertinentes de solución para satisfacer sus necesidades.” (Bosch y Bosch-Sijtsema, 2010b, c)

Lungu et al. (2010a) se presenta una definición diferente de los secos que es adoptada por una serie de documentos:

“Un ecosistema de software es un conjunto de proyectos de software que se desarrollan y evolucionan juntos en el mismo entorno.” (Lungu et al., 2010a)

Aunque son diferentes en semántica, pero al menos los ecosistemas de software incluyen las soluciones de software y las transacciones o relaciones entre ellos.

No es de extrañar, si atendemos a las definiciones que podemos ver que tienen dos cosas en común: software preocupación en alguna forma (los sistemas de software, productos, servicios, o una plataforma de software) y se están incluyendo todos algún tipo de relación, ya sea "simbiótica "," evolución común "," business "o" técnica ". Tomando las dos definiciones más amplias en perspectiva de Bosch et al. y Jansen, que se hace referencia en la mayoría de la literatura de los documentos que proporcionan una definición para SECO (65%), se pueden identificar tres elementos principales en sus definiciones:

Common Software: El software aparece ya sea como una "plataforma tecnológica común" (Jansen et al., 2009b), "soluciones de software" (Bosch, 2009) o "plataforma de software" (Bosch y Bosch-Sijtsema, 2010b, c).

Negocios: Este se expresa como "un conjunto de negocios" (Jansen et al., 2009b), "ecosistema empresarial" (Bosch, 2009), una "comunidad de usuarios que tienen necesidades que deben cumplir" (Bosch y Bosch-Sijtsema, 2010b, c). En este elemento, el término "Business" está implicando un sentido más amplio que los modelos de utilidad o ingreso. Este elemento también incluye posibles beneficios distintos de los ingresos financieros, por ejemplo, los beneficios de un actor se obtiene de la participación en un proyecto de código libre o abierto.

Conexión de Relaciones: "un conjunto de empresas (...) junto con las relaciones entre ellos" (Jansen et al., 2009b), "los actores del ecosistema social asociado" (Bosch,

2009), "la comunidad de expertos de dominio" y "comunidad de usuarios "(Bosch y Bosch-Sijtsema, 2010b, c).

2.1 Arquitectura de Ecosistema de Software

Cuando el desarrollo de software global (Global Software Development, GSD), los SECO y la línea de producción de software (Software Production Line, SPL) son colocados juntos en una arquitectura, desde un punto de vista social y de negocios, muchas organizaciones juegan con el modelo resultante de antiguos y nuevos procesos de ingeniería de software en el mercado.

2.1.1 SECO software engineering

Un aspecto importante es la arquitectura de software de un SECO, que debe apoyar la naturaleza del ecosistema (es decir, debe adaptarse a las necesidades de la SECO específica), siguiendo la gestión, normas y restricciones comerciales de la SECO y permitir la integración y la existencia de múltiples funcionalidades en un entorno seguro y de manera fiable.

Una arquitectura modular y flexible permitiría la integración y la interoperabilidad del software desarrollado (Viljainen y Kauppinen de 2011 y Bosch, 2009).

La estabilidad y la transparencia de las interfaces de la plataforma son esenciales para la integración de los componentes y la interacción (Cataldo y Herbsleb de 2010 y Bosch, 2010a). Los cambios en las interfaces o componentes existentes pueden dar lugar a incoherencias en los componentes dependientes (Robbes y Lungu, 2011, Lungu et al., 2010a y Lungu et al., 2010b).

Por otro lado, los enfoques centrados en el proceso no son eficaces en el manejo de software a gran escala, en lugar de la arquitectura del sistema debe ser utilizado como un mecanismo de coordinación (Bosch y Bosch-Sijtsema, 2010a). Y el Software en constante evolución requiere de la adaptación de los procesos de desarrollo de software. El desarrollo debe estar centrado en la integración, despliegue y comunicados independientes, y deben organizarse en una agrupación de liberación (Bosch y Bosch-Sijtsema, 2010b y Bosch, 2010a).

Estos principios se utilizan en la definición de la arquitectura de software de SECO (Kazman et al., 2012).

Además de la arquitectura de software, en los temas relacionados con ingeniería de software (SE), la obtención de requisitos aparece como un reto interesante en el concepto SECO como los actores son múltiples y distantes de la dirección central del ecosistema. Se propone el uso de la "cadena de valor" para propagar requisitos (Fricker, 2009, 2010).

2.1.2 SECO business and management

Esta categoría incluye publicaciones relacionadas con el negocio, los aspectos organizativos y de gestión de SECOS. Independientemente de cómo cada SECO está organizado, no es una entidad de organización y de gestión que se encarga de la supervisión, operación y toma de decisiones parte de la SECO ya ser una empresa de propiedad, una comunidad de código abierto o un híbrido de los dos. Esta categoría se subdivide en dos grupos: organización y gestión y negocio.

2.2 Actores Involucrados en Ecosistema de Software

La combinación de las definiciones anteriores con los tres elementos identificados, se define un ecosistema de software como la interacción de un conjunto de actores en la parte superior de una plataforma tecnológica común que resulta en una serie de soluciones de software o servicios.

Cada actor es motivado por un conjunto de intereses o modelos de negocio y comunicada con el resto de los actores y el ecosistema en su conjunto con las relaciones simbióticas, mientras que la plataforma tecnológica está estructurado de una manera que permita la participación y la contribución de los diferentes actores.

En otras palabras, la SECO ofrece posibilidades para que los actores se beneficien de su participación en el ecosistema. Los tipos de beneficios pueden variar según el actor y la naturaleza del ecosistema.

Las relaciones entre los actores en un SECO, pueden ser de interés comercial, donde los actores pudieran ganar regalías directas, por ejemplo: desarrolladores que han realizado aplicaciones para el iPhone y son vendidas en la App Store. Por otro lado, en un ecosistema no comercial los actores pudieran participar para obtener beneficios no monetarios (fama, conocimiento, ideología, etc.), un ejemplo de esto, son los desarrolladores que contribuyen con Apache.

Las relaciones entre los actores de un SECO, está caracterizada por un amplio espectro de relaciones simbióticas. Las cuales dependen de su actividad, los actores deberían tener beneficios mutuos (mutualismo), estar en competición directa (competición/antagonismo), no ser afectados (neutralismo) o uno permanecer sin efectos mientras el otro se está beneficiando (comensalismo) o recibiendo daño (parasitismo) por su relación.

De hecho, literatura de software de código abierto (OSS) refuerza la importancia de la colaboración entre actores para poder desarrollar una plataforma de código abierta bien madura.

Olavo Barbosa et. al, A Systematic Mapping Study on Software Ecosystems

Se está prestando cada vez más atención a la conectividad y la dependencia en las relaciones entre las empresas. Las innovaciones no se originan en una sola organización, sino son co-innovación de los diferentes actores. Empresas capacidades de co-evolucionan en torno a una nueva innovación: trabajan cooperativamente y competitiva para apoyar nueva productos, satisfacer las necesidades del cliente, y, finalmente, incorporar la siguiente ronda de innovaciones. Estas redes informales de proveedores, distribuidores, outsourcing empresas, desarrolladores de productos o servicios relacionados, proveedores de tecnología, y una serie de otras organizaciones afectan y son afectados por, la creación y entrega de las ofertas propias de la empresa.

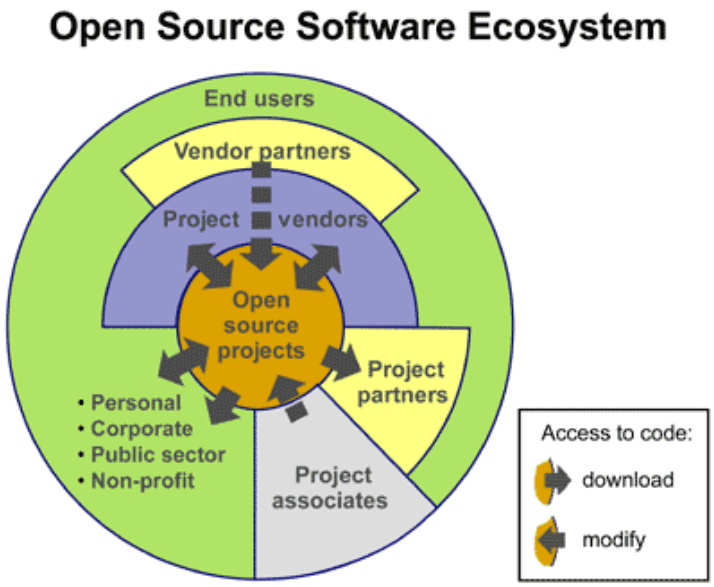

Figura 1. Ecosistema de software open source

Una plataforma tecnológica abierta en combinación con un conjunto de procesos de gestión y modelos de negocio, no puede crear una SECO sin el aspecto social. Una comunidad, red social o un conjunto de actores tejido en torno a una plataforma y conjuntos de reglas que se comunican e interactúan entre sí y con la plataforma es esencial. Debido a la existencia de esta interacción, la arquitectura de software de la plataforma tiene que ser diseñado con diferentes consideraciones que una plataforma propietaria. El proceso de gestión, modelos de negocio y las cuestiones de derechos de propiedad intelectual son más complicados, mientras que al mismo tiempo la evolución del sistema es más rápido y hacía varias direcciones mientras que las ganancias de posición de un SECO es privilegiada en el mercado. Hay varios agentes que podrían ser parte de una SECO. La siguiente lista ofrece una visión general de los actores más comunes:

Orchestrator, es la piedra angular que organiza, modela, gestiona, o es el dueño de la plataforma, también es una empresa, un departamento de una empresa, actor o grupo de actores, comunidad o entidad independiente que se encarga de el buen funcionamiento de la SECO. Esta unidad es típicamente el manejador del SECO ejecutando la plataforma, la creación y aplicación de normas, procesos, procedimientos comerciales, el establecimiento y supervisión de las normas de calidad y / o instrumentar las relaciones actores SECO.

Niche, es el actor SECO que contribuye a la SECO por lo general el desarrollo de o la adición de componentes de la plataforma, produciendo funcionalidad que los clientes requieren. Este actor es parte de la SECO y complementa el trabajo de la piedra angular, proporcionando valor al ecosistema. Dependiendo del modelo de gestión del ecosistema de los nichos de mercado pueden influir en la toma de decisiones en la gestión de la SECO decisión.

External Actor, es el actor (empresa, persona, entidad) que hace uso de las posibilidades que ofrece el ecosistema y proporcionar así un valor indirecto a favor del ecosistema. Este actor es externo a la gestión SECO y suele tener una actividad limitada a los intereses del actor. Dependiendo de la naturaleza del ecosistema, el actor externo puede ser el desarrollo de la parte superior o en paralelo a la plataforma SECO, identificar los errores, promover la SECO y sus productos o proponer mejoras.

Vendor, es principalmente la empresa o unidad de negocio que hace que el beneficio de la venta de los productos de la SECO a los clientes, usuarios finales u otros vendedores / distribuidores de valor añadido. Los productos pueden ser integraciones completas, componentes, la venta o el arrendamiento de licencias o acuerdos de apoyo. Un vendedor que está modificando el producto SECO mediante, por ejemplo, la adición de la funcionalidad o la combinación de diferentes componentes entre sí se llama distribuidor del valor añadido (VAR).

Costumer, o "usuario final" es la persona, empresa, entidad que ya sea compra u obtiene un producto completo o parcial de la SECO, o un actor de nicho, ya sea directamente de la SECO desde el nicho de mercado o a través de un vendedor VAR.

2.3 Principales Características de un Ecosistema de Software

Como una nueva perspectiva en ingeniería de software, los SECOs deberían caracterizarse con el fin de entender las tendencias y el impacto en la ingeniería de software y analizar el conocimiento en esa área.

Unas de las principales características de los ecosistemas de software identificados en algunos estudios afirman que los ecosistemas de software están vinculados a los ecosistemas naturales, las líneas de productos de software, y los ecosistemas de

negocios. Estos campos son considerados por muchos autores como los orígenes de la investigación de los ecosistemas de software.

- Heredar características de los ecosistemas naturales, como el mutualismo, comensalismo, amensalismo, simbiosis y así sucesivamente.
- Está vinculado a la perspectiva de ecosistema empresarial.
- Está vinculado a los conceptos arquitectónicos como la estabilidad de la interfaz, la gestión de la evolución, la seguridad, la fiabilidad.
- Está vinculado al modelo de desarrollo de Lineas de producción de software (SPL).
- Está vinculado al Modelo Open Source Development.
- Puede ser utilizado para negociar los requisitos que alinean las necesidades con soluciones, componentes y carteras.
- El desarrollo en colaboración con el gobierno puede ser visto como un tipo de ecosistema digital.
- Estar relacionado con los procesos de innovación.
- Tener impacto en las pequeñas y medianas empresas.

2.4 Beneficios, Desafíos y Limitaciones de un Ecosistema de Software

Entre los beneficios de los SECOs se pueden mencionar:

- El fomento de la co-evolución.
- La innovación dentro de la organización.
- El incremento del atractivo para nuevos actores.
- Disminuye los costos involucrados en el desarrollo y distribución de software.
- Ayuda a analizar y comprender la arquitectura de software con el fin de decidir qué plataforma usar.
- Apoya la cooperación y el intercambio de conocimientos entre las múltiples e independientes entidades.
- Permite el análisis de los requisitos de comunicación entre las partes interesadas.
- Viene como alternativa para superar los retos durante el diseño y mantenimiento de aplicaciones distribuidas.
- Proporciona ayuda a las tareas de identificación de negocios, diseño de la arquitectura del producto, y la identificación de riesgos.
- Proporciona información para el administrador de la línea de productos en relación con las dependencias de software.
- Sostenibilidad tecnológica.

Entre los principales desafíos y limitaciones involucrados en las perspectivas de los SECOs se puede mencionar:

- El establecimiento de relaciones entre los actores del ecosistema y la propuesta de una representación adecuada de las personas y sus conocimientos en la modelización de ecosistemas.
- Desafíos arquitectónicos clave como: la estabilidad de la interfaz de la plataforma, la gestión de la evolución del software, la seguridad, la fiabilidad, la forma de apoyar la estrategia de negocio, arquitecturas adecuadas para apoyar el desarrollo del estilo de código abierto.
- La heterogeneidad de las licencias de software y la evolución de sistemas en un ecosistema. Las organizaciones deben manejar estos temas con el fin de disminuir los riesgos de dependencia.
- Obstáculos técnicos y socio-organizativos para la coordinación y la comunicación de los requisitos de los proyectos distribuidos geográficamente.
- Infraestructura y herramientas para fomentar la interacción social, la toma de decisiones y el desarrollo a través de las organizaciones que participan tanto de código abierto y de los ecosistemas propios.

2.5 Principales Áreas de Interés de un Ecosistema de Software

Las principales áreas de estudio están dirigidas a involucrar la evolución del software y la co-innovación. Estos hechos son propiedades esenciales que parecen tener una implicación vital para el desarrollo de la industria. Por último se pueden mostrar las áreas donde están concentrados los estudios de los ecosistemas de software, como se observa en la figura 2.

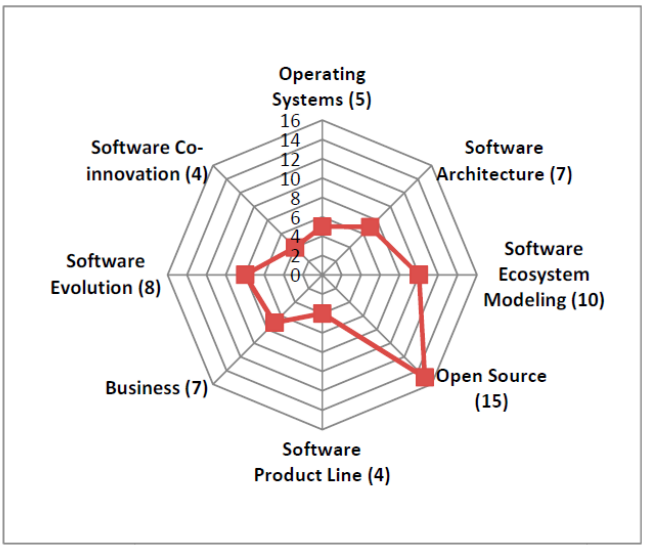

Figura 2. Áreas de mayor investigación de los SECO´s

Ventajas de los SECO´s

La infraestructura es compartida y la proporcionan los usuarios. Escalabilidad: conforme hay más usuarios no se necesitan más recursos. No existe un tecno punto de control. Iguales oportunidades de acceso a la infraestructura y posibilidad de modificarla y hacerla evolucionar. Independencia de la plataforma de software y hardware. Interoperabilidad y la utilización de la información y los servicios existentes en formatos abiertos.

2.6 Modelo de clasificación de Ecosistemas de Software

Ya que se definió a los ecosistemas de software como parte de un ecosistema de negocio. Se pueden usar modelos de clasificación de estos últimos. En los modelos de negocio se identifican factores como: ciclo de vida del producto, área geográfica y ecosistemas diversos. Para el manejo de este modelo se definen específicamente cuatro factores que pueden ser observados en la figura siguiente.

Tecnología Base: Por definición de ecosistema de software existirá una tecnología que soporte a un ecosistema de software, entre los tipos de tecnología podríamos establecer: plataformas de software, plataforma de servicio de software y software estándar. Muchos de los ecosistemas de software están soportados por una o más plataformas de software.

La plataforma de servicio de software s aquella donde se ofrece un servicio en línea en torno al cual otros ecosistemas participantes pueden reunirse y no pueden tener el servicio de manera individual en los clientes. Por último, un ecosistema puede basarse en un estándar de software, tales como el ecosistema XBRL o la Alianza Open Design, que basa la mayor parte de su tecnología alrededor de los estándares DWG de Autodesk.

Coordinadores: los coordinadores son aquellos que tienen la mayor influencia sobre el ecosistema, los propietarios tienen control sobre las herramientas y métodos que incrementan el éxito del ecosistema. El ecosistema de software puede ser propiedad de la comunidad o una agrupación privada. Un ejemplo de Ecosistema de propiedad de la comunidad es Eclipse. Los deseos son representados por un consorcio. En el caso de propiedad privada se tiene mayor control sobre los elementos y herramientas, tal es el caso por ejemplo de IOS en el caso de Apple.

Mercados de extensión: Muchos de los ecosistemas de software se centran alrededor de extensiones del mercado, conocidas como las app store y app market. Un ecosistema de software puede no tener una extensión de mercado, solo una lista de extensiones, una extensión de mercado comercial o múltiples extensiones de mercado.

Cuando un ecosistema no tiene una extensión de mercado explícita, los componentes pueden ser conocidos por medio de terceros. Un ejemplo de ello es AutoCAD que tiene una pequeña lista de componentes para la comunidad pero existen muchas más extensiones solo asequibles por medio de terceros.

Accesibilidad: Este factor es el que determina la facilidad de unirse a un ecosistema, las barreras que existen para entrar, y la forma en que se determina quién entra. Para los ecosistemas de software existen tres tipos de accesibilidades: open source, screened but free, y paid. En la primera, open source es un ecosistema donde se puede hacer contribuciones, crear y publicar componentes sin ninguna limitante. El de pago como en IOS es aquella donde se paga para poder desarrollar aplicaciones para el ecosistema.

Figura 3. Factores para el modelo de clasificación

Ahora bien con los cuatro factores ya definidos se puede realizar la clasificación de los ecosistemas por medio de una sentencia como la siguiente:

"El [NOMBRE] ecosistema de software es basado en una {plataforma de software, plataforma de servicio de software, software estándar} y es coordinada por {entidad privada, comunidad} con {sin extensión de mercado, una lista de extensiones, una extensión de mercado, una extensión de mercado comercial, múltiples extensiones de mercado} en la cual los participantes pueden crear extensiones {gratis, después de observación, pagando}"[4].

El uso de estos cuatro factores es solo un modelo de clasificación pueden realizarse variaciones y agregar nuevos factores y características como: "efecto en la red", "multi-homing", "costes de cambio" entre otras varias, es por esto que es un modelo de aproximación con solo cuatro principales o más relevantes.

3 Ecosistemas Open Source

3.1 Análisis de la evolución sobre los aspectos sociales en los ecosistemas de software de código libre

La mayoría de estudios empíricos sobre la evolución de software de código abierto se centran en aspectos técnico, tales como objetos de código fuente. Estos estudios ignoran en gran parte los aspectos sociales, por ejemplo, el impacto que tienen en el usuario y las comunidades de desarrolladores (y su interacción) sobre la evolución del proyecto del software.

Cambios importantes en la comunidad (tales como la inesperada salida de una persona clave, la adquisición del proyecto de una nueva comunidad) o en el producto de software (por ejemplo, una reestructuración, reemplazo o adición de una parte sustancial de la base del código) pueden influir significativamente en la manera que el proyecto de software continuará evolucionando con el tiempo.

Una mejor comprensión de este impacto, nos permitirá a mediano plazo, idear modelos de predicción, directrices y mejores prácticas que permitan a las comunidades mejorar sus herramientas y prácticas actuales que pueden utilizarse para controlar y mejorar sus procesos de trabajo actual, para comunicar efectivamente, y para que el software sea más atractivo para los desarrolladores y usuarios.

3.1.1 Aspectos sociales

La comunidad de desarrolladores de un proyecto de software está compuesta por personas que crean y modifican objetos de software. La buena comunicación es un factor fundamental de éxito para cualquier proyecto. Esto es especialmente cierto para los proyectos de código abierto debido a que a menudo se desarrollan de manera geográficamente distribuida.

Para este tipo de proyectos es también, en la mayoría de los casos, más fácil que se involucren en el equipo de desarrollo, lo que implica que la estructura del equipo debe ser más flexible para integrar a recién llegados y para afrontar a salida frecuente de los desarrolladores.

Los proyectos de código abierto se basan en una serie de herramientas que acceden por la comunidad de desarrolladores para compartir e intercambiar información, comunicarse y coordinar su trabajo. En la práctica, tas herramientas hacen uso intensivo de Internet. Las principales herramientas utilizadas por estas comunidades son los sistemas de control de versiones (como Subversión o Git), sistemas de seguimiento de error (tal como Bugzilla), listas de correo y foros de desarrolladores.

El software de código abierto suele ser obra de personas con mucho talento o de grupos informales o "comunidades" de programadores que desean resolver un problema técnico y compartir sus resultados con el resto del mundo. Algunos de sus defensores declaran que los modelos de código abierto generan más innovación, mientras que otros afirman que permiten producir modelos iguales o incluso

superiores a los productos comerciales de la competencia. Debido a su forma de distribución, puede existir una gran variedad de "productos" resultantes de un programa de código abierto determinado, e incluso pueden haberse derivado de él muchos otros programas poco conocidos con comportamientos algo o muy diferentes.

Los programas comerciales o privados suelen diseñarse y desarrollarse para responder a la demanda del mercado, o al menos a una necesidad percibida en el mercado. Sus características suelen determinarlas el mercado y los usuarios. Su desarrollo se realiza de forma relativamente estructurada y disciplinada, y los productos resultantes suelen estar bastante bien catalogados, ser de calidad comprobada y contar con buen apoyo.

Las características de los programas comerciales suelen ser el resultado de una larga evolución basada en el mercado. Evidentemente, el usuario es quien paga todo esto, lo quiera o no.

3.2 Licencias open source

Una licencia de código abierto es una licencia de software que permite que tanto el código fuente como los archivos binarios sean modificados y redistribuidos libremente y sin tener que pagar al autor original. Sin embargo, ciertas licencias de código abierto pueden incorporar algunas restricciones, como el requisito de mantener el nombre de los autores y la declaración de derechos de autor en el código, o permitir la modificación del código sólo para usos personales o la redistribución del software para usos no comerciales. Un grupo popular de licencias de software de código abierto son aquellas aprobadas por la Open Source Initiative (OSI) basándose en su Open Source Definition.

3.2.1 Elementos de una licencia de software

El licenciante o proveedor-licenciante: es aquel que provee el software más la licencia al licenciatario, la cual, le permitirá a este último tener ciertos derechos sobre el software. El rol de licenciante lo puede ejercer cualquiera de los siguientes actores: el autor o autores que desarrollaron el software, el titular de los derechos de explotación, o un distribuidor de software.

Garantía de titularidad: ofrecida por el licenciante o propietario, en la cual, asegura que cuenta con suficientes derechos de explotación sobre el software como para permitirle proveer una licencia al licenciatario.

El licenciatario o usuario-licenciatario: es aquella persona física o jurídica que se le permite ejercer el derecho de uso más algún otro derecho de explotación sobre un determinado software cumpliendo las condiciones establecidas por la licencia otorgada por el licenciante.

3.2.2 Clasificación de las licencias open source

Licencias permisivas: Estas licencias no imponen prácticamente ninguna condición sobre quien recibe el software, y sin embargo le dan permiso de uso, redistribución y

modificación. De hecho, bajo esta licencia se permite la redistribución del software bajo una licencia comercial o privativa. Algunas de las licencias pertenecientes a esta clase son: Apache Software License, BSD License, Academic Free License, Perl License, etc.

Licencias restrictivas o robustas y con copyleft: Estas licencias intentan garantizar siempre los derechos de los usuarios, y por la tanto bajo esta licencias se logra mantener la libertad del código en todo momento, no permitiendo modificaciones que cambien el código a una licencia privativa o comercial. El principal ejemplo de este tipo de licencia es la Licencia GPL, eCos License, Affero GPL, etc.

Licencias robustas débiles: Estas licencias son débiles, con copyleft débil/suave o híbridas, que contienen una cláusula que obliga a que las modificaciones que se realicen al software original se deban licenciar bajo los mismos términos y condiciones de la licencia original, pero que las obras derivadas que se puedan realizar de él puedan ser licenciadas bajo otros términos y condiciones distintas. Como ejemplo de este tipo de licencia existen: GNU LGPL, MPL, Eclipse Public License, etc.

3.2.2.1 General Public License (GPL)

Una de las más utilizadas es la Licencia Pública General de GNU (GNU GPL). El autor conserva los derechos de autor (copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del software permanecen bajo los términos más restrictivos de la propia GNU GPL. Esto hace que sea imposible crear un producto con partes no licenciadas GPL: el conjunto tiene que ser GPL.

Es decir, la licencia GNU GPL posibilita la modificación y redistribución del software, pero únicamente bajo esa misma licencia. Y añade que si se reutiliza en un mismo programa código "A" licenciado bajo licencia GNU GPL y código "B" licenciado bajo otro tipo de licencia libre, el código final "C", independientemente de la cantidad y calidad de cada uno de los códigos "A" y "B", debe estar bajo la licencia GNU GPL.

En la práctica esto hace que las licencias de software libre se dividan en dos grandes grupos, aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL (y que inevitablemente desaparecerán en el proceso, al ser el código resultante licenciado bajo GNU GPL) y las que no lo permiten al incluir mayores u otros requisitos que no contemplan ni admiten la GNU GPL y que por lo tanto no pueden ser enlazadas ni mezcladas con código gobernado por la licencia GNU GPL.

3.2.2.2 Affero General Public License (AGPL)

La Licencia Pública General de Affero (en inglés Affero General Public License, también Affero GPL o AGPL) es una licencia copyleft derivada de la licencia GNU GPL, diseñada específicamente para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red. La AGPL es íntegramente una GNU GPL con una cláusula nueva que añade la obligación de distribuir el software si éste se ejecuta para ofrecer servicios a través de una red de ordenadores.

La Free Software Foundation recomienda que el uso de la GNU AGPLv3 sea considerado para cualquier software que usualmente corra sobre una red.

3.2.2.3 Licencias Berkeley Software Distribution (BSD)

La licencia BSD, es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). El autor, bajo tales licencias, mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario. Son muy permisivas, tanto que son fácilmente absorbidas al ser mezcladas con la licencia GNU GPL con quienes son compatibles.

Puede argumentarse que esta licencia asegura "verdadero" software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al software, y que puede decidir incluso redistribuirlo como no libre. Otras opiniones están orientadas a destacar que este tipo de licencia no contribuye al desarrollo de más software libre (normalmente utilizando la siguiente analogía: "una licencia BSD es más libre que una GPL si y sólo si se opina también que un país que permita la esclavitud es más libre que otro que no la permite").

3.2.2.4 Mozilla Public License (MPL)

La Licencia Pública de Mozilla (en inglés Mozilla Public License o MPL) es una licencia de código abierto y de software libre. Fue desarrollada originalmente por Netscape Communications Corporation (una división de la empresa América Online), y más tarde su control fue traspasado a la Fundación Mozilla.

La licencia MPL cumple completamente con la definición de software de código abierto de la Open Source Initiative (OSI) y con las cuatro libertades del software libre enunciadas por la Free Software Foundation (FSF). Sin embargo la MPL deja abierto el camino a una posible reutilización no libre del software, si el usuario así lo desea, sin restringir la reutilización del código ni el re-licenciamiento bajo la misma licencia.

3.2.2.5 Apache Software License (ASL)

La licencia Apache (Apache License o Apache Software License para versiones anteriores a 2.0) es una licencia de software libre creada por la Apache Software Foundation (ASF). La licencia Apache requiere la conservación del aviso de copyright y el disclaimer, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

La Licencia Apache permite al usuario del software, la libertad de usarlo para cualquier propósito, distribuirlo, modificarlo, y distribuir versiones modificadas de ese software. Además no exige que las obras derivadas (versiones modificadas) del software se distribuyan usando la misma licencia, ni siquiera que se tengan que distribuir como software libre/open source. Sólo exige que se mantenga una noticia que informe a los receptores que en la distribución se ha usado código con la Licencia Apache.

3.2.2.6 Eclipse Public License (EPL)

Es una licencia utilizada por la Fundación Eclipse para sus proyectos de software. Esta licencia sustituye la Licencia Publica Común (CPL) y elimina condiciones relacionadas con los litigios de patentes.

Está diseñada para ser una licencia de software favorable a los negocios y cuenta con disposiciones más débiles que las licencias copyleft, y como la Licencia Pública General de GNU (GPL). El receptor de programas licenciados EPL pueden utilizar, modificar, copiar y distribuir el trabajo y las versiones modificadas, en algunos casos están obligados a liberar sus propios cambios.

3.2.2.7 Copyleft

El copyleft se practica al ejercer el derecho de autor que consiste en permitir la libre distribución de copias y versiones modificadas de una obra u otro trabajo, exigiendo que los mismos derechos sean preservados en las versiones modificadas. La efectividad de ejercerlo puede depender de la legislación particular de cada país, pero en principio se puede utilizar para programas informáticos, obras de arte, cultura, ciencia, o cualquier tipo de obra o trabajo creativo que sea regido por el derecho de autor.

El término surge en las comunidades de software libre como un juego de palabras en torno a copyright: "Derecho de autor", en inglés (literalmente: "Derecho de copia") con otro sentido, el de left: pretérito del verbo dejar o permitir. Se considera que una licencia libre es copyleft cuando además de otorgar permisos de uso, copia, modificación y redistribución de la obra protegida, contiene una cláusula que dispone una licencia similar o compatible a las copias y a las obras derivadas.

3.3 Modelos de negocio para el software open source

Un modelo de negocio describe la lógica de cómo una organización crea, entrega y captura valor (las formas económicas, sociales, culturales o de otro tipo de valor). El proceso de construcción del modelo de negocio es parte de la estrategia de negocio.

En la industria del software hay mucha actividad alrededor del código abierto, así como las inversiones en proyectos de código abierto y su creciente uso en general. Aceptando la coexistencia del software libre de código abierto y el software comercial de código abierto, surge la pregunta sobre ¿cómo se pueden articular modelos de negocio que permitan ganar dinero a las empresas o individuos sobre la base del código abierto?.

Para una empresa que de software comercial, el software de código abierto se licencia a la empresa bajo una licencia de código abierto o es un software proporcionado por la empresa a los usuarios bajo una licencia de código abierto.

A continuación se presentan los enfoques de negocios comerciales existentes y legales en el contexto de las licencias de código abierto de software de código abierto.

3.3.1 Modelo de negocio por licenciamiento dual

El uso de la doble licencia proporciona una oferta de software bajo una licencia de código abierto, sino también en los términos de licencia de propiedad independiente. La versión patentada se puede vender para financiar el desarrollo continuado de la versión libre de código abierto. Los clientes pueden ser atraídos por una versión de software de código abierto sin costo y, posteriormente, ser atraídos por una versión comercial de software de código abierto, con más funcionalidades, soporte, o la capacidad de generar productos nuevos.

3.3.2 Modelo de negocio de venta de soporte, formación y consultoría

El rendimiento financiero de los gastos en software de código abierto también puede provenir de la venta de servicios, tales como:

Proveedores de Soporte: son conocedores expertos de un producto, se especializan en dar soporte (help-desk, soporte funcional) a los usuarios de dichos productos.

Formadores: conocedores expertos de un producto, desarrollan actividades de formación para los usuarios finales, de dichos productos.

Proveedores servicios: que ofrecen el servicio comercial de compilación y empaquetado de software. Además, proporciona las mercancías físicas como medio de instalación (por ejemplo, DVD) puede ser un servicio comercial.

Compañías de código abierto que se utilizan con éxito este modelo de negocio son, por ejemplo RedHat y IBM.

Algunas organizaciones de código abierto, por ejemplo Mozilla o de la Fundación Wikimedia, tratar de vender artículos de marca (por ejemplo, T-Shirts, tazas de café, etc.) lo que puede ser visto también como un servicio adicional a la comunidad de usuarios.

Servicios de consultaría: sirven para planear y ejecutar las instalación de los entornos del sistema que contiene el software.

Evangelizadores: son aquellos líderes del movimiento código abierto que se convierten en gurús y pueden cobrar por dar conferencias, escribir libros o artículos, textos para cursos, etc.

3.3.3 Modelo de negocio de venta de software como servicio

Vender suscripciones de cuentas online, para que los clientes tengan acceso a los servidores de soporte, es una forma de obtener beneficios basados en software de código abierto. Siendo la combinación del software y del servicio un plus para este modelo de negocio. Otra modalidad es proporcionar servicios de cloud computing o software como un servicio (SaaS), sin la liberación del propio software de código abierto, ni el binario, ni en forma de código fuente, este forma se ajusta a la mayoría de licencias de código abierto (con excepción de la AGPL).

3.3.4 Modelo de negocio con financiamiento de empresas

Este modelo se desarrolla de muchas formas distintas, y con mecanismos para conseguir recursos que varían muchísimo de un caso a otro. Cada proyecto tiene su propia forma de financiarse, desde el que está formado completamente por desarrolladores voluntarios, hasta el que es llevado a cabo por una empresa, gobiernos, universidades, u otras organizaciones no gubernamentales.

El financiamiento público es entidad que puede ser directamente un gobierno (local, regional, nacional o incluso supranacional) o una institución pública (por ejemplo, una fundación). Normalmente, la entidad financiadora no busca recuperar la inversión (o al menos no de forma directa), aunque suele tener objetivos claros (favorecer la creación de tejido industrial e investigador, promover cierta tecnología o cierto tipo de aplicaciones, etc.).

Financiación por quienes necesitan mejoras sobre la base de un producto ajeno, el cual lo mejoran o adaptan (soluciones sectoriales, adaptaciones a legislaciones nacionales, etc.) posicionándose como unos "segundos productores".

Financiación con inversión interna de la misma empresa, que desarrolla su propio software como parte de su modelo de negocio. La idea es invertir en un proyecto de código abierto con la idea de rentabilizar posteriormente la inversión, siendo los beneficios relacionados las ventajas que obtenga la empresa por la producción de software de código abierto.

3.3.5 Modelo de negocio por donaciones

El modelo de negocio por donaciones es impulsada por la comunidad de usuarios, para financiar el desarrollo de software de código abierto, por ejemplo, SourceForge permite a los usuarios donar dinero a proyectos que tienen alojados elegido en la pestaña donate y aceptar la donación. Estas donaciones son realizadas con tarjetas de visa, mastercard, american express, discover y por PayPal,

Además, existen grandes campañas de donación. Por ejemplo, en 2004 la Fundación Mozilla había una campaña de recaudación de fondos para apoyar el lanzamiento de Firefox 1.0 navegador web. Un anuncio de dos páginas en la edición del 16 de diciembre New York Times, colocados por la Fundación Mozilla, en coordinación con la extensión de Firefox, aparece el nombre de los miles de personas en el mundo que donaron a la campaña.

3.3.6 Modelo de negocio con software publicitario

Con el fin de comercializar software libre, muchas empresas (incluyendo Google, Mozilla y Ubuntu) se han movido hacia un modelo económico de software relacionado con publicidad. Por ejemplo, la aplicación de código abierto AdBlock Plus es pagada por Google para dejar una lista blanca de anuncios.Otro ejemplo es Sourceforge, un proveedor de servicios de proyecto de código abierto, tiene el modelo de ingresos de las ventas de publicidad de la bandera en su página web. En 2006, Sourceforge Inc informó recaudaciones trimestrales de USD $ 6.5 millones y $ 23 millones en 2009.

3.3.7 Modelo de negocio de venta opcional de extensiones propietarias

Algunas compañías venden opcionalmente extensiones propietarias, módulos, plugins o add-ons, para un producto de software de código abierto. Esto puede ser una licencia que se ajusta a un enfoque con muchas licencias de código abierto, si se hace técnicamente con mucho cuidado. Por ejemplo mezclar código propietario y código de licencia de código abierto con las bibliotecas vinculadas estáticamente o compilar todo el código fuente junto en un producto de software, puede violar las licencias de código abierto, por lo tanto, mientras que se mantengan separadas por interfaces y el enlace a las bibliotecas sea dinámico, pueden a menudo conformarse la licencia.

3.3.8 Modelo de negocio re-licencia bajo una licencia propietaria

Si un producto de software sólo utiliza software propio y el software de código abierto bajo una licencia de software libre permisiva, una empresa puede volver a adquirir la licencia del producto de software resultante bajo una licencia propietaria y vender el producto sin el código fuente o libertades del software. Por ejemplo, Apple Inc. es ávido usuario de este enfoque mediante el uso de código fuente y forma varios proyectos de software de código abierto, por ejemplo, desde el sistema operativo FreeBSD que tiene un kernel bajo licencia BSD era utilizado en los Mac PCs que se venden como productos propietarios.

3.3.9 Modelo de negocio retrasando la liberación de código abierto

Este modelo se basa en poner a disposición de los usuarios de pago la última versión de software de código abierto comercial, y después de un período de tiempo determinado los parches son liberados para el resto de usuarios que utilizan la versión no comercial. Esto se realiza hasta que se considera que se ha recuperado la inversión de desarrollo del proyecto, teniendo como resultado la versión libre y completa.

3.3.10 Conclusión de un modelo de negocio de código abierto

En general, el modelo de negocio de código abierto hace más sencillos todos los aspectos de una empresa de software porque, sin tener que ocultar tu propiedad intelectual, puedes incorporar las grandes ideas de todos los interesados, puedes hacer que te ayuden en el desarrollo, en el control de calidad y en la propagación de tu producto o proyecto.

El reto del modelo de código abierto, para una empresa comercial de código abierto, es mantener el equilibrio entre la comunidad y los clientes suscriptores. Porque realmente, el cliente suscriptor sólo se va a suscribir si ve valor en ello, y siempre habrá gente en la comunidad que no se suscribirá nunca. Y se debe mantener el equilibrio correcto porque sin los clientes suscriptores no habría empresa y la comunidad perdería el principal motor para el desarrollo de los proveedores de código abierto.

3.4 Comunidad Open Source como proveedores

Casi cualquier software comercial contiene componentes que se encuentran bajo la licencia de código abierto o utilizan software de código abierto como entorno de ejecución. Las principales razones para utilizar código abierto como recurso

suministrado es la calidad y las ventajas de coste. Por lo tanto al proveedor de software, le viene bien el código abierto y provee de significantes ventajas en relación al coste y comparación con la funcionalidad de software similar, y es propiedad de la programación desde cero.

El proveedor de software también puede incluir y enviar el software de código abierto junto con sus soluciones, ya que cada software de código abierto incluye términos de la licencia y el proveedor debe cumplir dicho términos de la licencia de código abierto utilizado. Si el proveedor de software incluye en el software de código abierto, que es responsable del soporte y mantenimiento del software suministrado. Entonces el proveedor de software tiene que asegurarse de que es capaz de mantener y apoyar el software de código abierto.

Una empresa especializada en el análisis de uso de código abierto y en el análisis de los términos de licencia adjunta, es Black Duck Software. Ofrecen una herramienta que analiza el código fuente y el software de código abierto utilizado. Entonces es posible determinar si se puede cumplir con los términos de licencia y si desea seguir utilizando el software de código abierto.

3.5 Comunidad open source propietaria, mantenedora y de soporte

El proveedor de software podría decidir donar el código fuente a la comunidad y dejar que la unidad de innovación de productos de la comunidad se encargue de ello, así como las actividades de mantenimiento. Al hacer esto, el proveedor de software reduce más su costo para desarrollar y mantener software.

Los proveedores de software consideran algunas de las siguientes razones para la donación de software para la comunidad e código abierto, como por ejemplo:

- Crear una visibilidad de experiencia de compañía de software: esto es especialmente interesante para las pequeñas empresas que deseen ganar visibilidad y reputación dentro de una comunidad más grande de experto en la materia.
- Para eliminar el coste por desarrollo de producto/software, mantenimiento y soporte: si algún producto está en las últimas etapas de su ciclo de vida o género muy pocas ventas.

3.6 Comunidad open source como canal de ventas

Los miembros de la comunidad avalan y recomiendan su uso (a las empresas). Las empresas pueden elegir entre el open source y la licencia comercial del software. Si elige la licencia comercial, será proporcionada por el proveedor de software. Dependiendo de la licencia open source elegida, el proveedor de software puede forzar a los clientes para adquirir una licencia comercial. Este es el caso de las licencias open source, que no permiten el uso comercial del software bajo la licencia open source. Otro caso es el cliente quiere hacer cambios de propiedad sobre el software y mantener la propiedad del software, que entra en conflicto con licencias copyleft. Las licencias

copyleft exigen que todas las versiones, incluyendo modificado y versiones extendidas, estén disponibles gratuitamente para la comunidad.

4 Apache Software Foundation (ASF)

4.1 Objetivo de la ASF

La Fundación de Software Apache, 501 EE.UU. (3) (c) es una organización sin fines de lucro, proporciona apoyo organizativo, legal y financiero para una amplia gama de más de 140 proyectos de software de código abierto. La Fundación ofrece un marco establecido para la propiedad intelectual y las contribuciones financieras al mismo tiempo que limita la exposición legal potencial para nuestros committers de proyecto.

A través de un proceso de desarrollo colaborativo y meritocrático conocida como "The Way Apache", los proyectos de Apache ™ ofrecen productos de software a nivel empresarial, de libre disposición que atraen a grandes comunidades de usuarios. La licencia Apache pragmática facilita a todos los usuarios, comerciales e individuales, para desplegar los productos de Apache.

Los proyectos de Apache se caracterizan por un proceso de colaboración en el desarrollo basado en el consenso, una licencia de software abierta y pragmática, y el deseo de crear un software de alta calidad que es líder en su campo. Además de ser no sólo un grupo de proyectos que comparten un servidor, sino una comunidad de desarrolladores y usuarios.

4.2 Meritocracia

La meritocracia es el principio de gobierno de la ASF que se basa en el mérito. Cada uno de sus miembros que conforman la ASF, han demostrado su eficacia y compromiso al contribuir con el desarrollo de software de código abierto en uno o más de los proyectos de la Fundación, para su evolución.

4.3 Estructura de la Fundación

La composición de la ASF se compone de las personas, y no de empresas. Los miembros tienen un interés jurídico en el ASF. La estructura de la fundación es influenciada por el espíritu de la comunidad, con transparencia, consenso, y meritocracia.

La ASF está constituida por comunidades, que separadas se denominan "Proyectos" y si bien son similares, cada uno de ellos mostraran pequeñas diferencias que los hacen especiales.

Con el fin de reducir la fricción y permitir surgir a la diversidad, en lugar de forzar un monocultivo de la parte superior, los proyectos se designan las organizaciones centrales de toma de decisiones del mundo Apache. Cada proyecto se delega a la autoridad sobre el desarrollo de su software, y se le da una gran libertad en el diseño de sus propios estatutos técnicos y sus propias reglas que rigen.

La Fundación se rige por las siguientes entidades:

- **Board of Directors** (junta directiva), que gobierna la fundación y se compone de los miembros.
- **Project Management Committees** (PMC, comité de gestión de proyecto), que rigen los proyectos, y se componen de committers. (Teniendo en cuenta que cada miembro es, por definición, también committer.)

4.4 Board of directors (junta directiva)

El Board of Directors de The Apache Software Foundation (ASF), es responsable de la gestión y la supervisión de los negocios y asuntos de la corporación de acuerdo con los Estatutos. Esto incluye la gestión de los activos de la empresa (fondos, propiedad intelectual, marcas registradas, y equipo de apoyo) y la asignación de recursos de la empresa a los proyectos. Sin embargo, la técnica de decisión sobre el contenido y la orientación de los proyectos de Apache se asigna a cada respectivo comité de gestión del proyecto. La Board of directors, es elegida anualmente por los miembros de la ASF.

Las reuniones de la Junta y las minutas son publicadas en el calendario de la junta. Muchos otros registros públicos de la corporación son publicados públicamente.

Tabla 2. Consitutción del Board of Directos de la ASF.

Cargo	Nombre del representante
Chairman (presidente)	Brett Porter
Vice Chairman (vicepresidente)	Greg Stein
President	Ross Gardler
Exec. V.P	Rich Bowen
Treasurer (tesorero)	Chris Mattmann
Assistant Treasurer (asistente de tesorero)	Sam Ruby
Secretary (secretario)	Craig Russell
Assistant Secretary (asitente de secretario)	James Carman
V.P., **Brand Management** (vicepresidente de gestión de marca)	Shane Curcuru
V.P., **Community Development** (vicepresidente comunidad desarrolladores)	Luciano Resende
V.P., **Fundraising** (vicepresidente recaudación de fondos)	Upayavira
V.P., **Infrastructure** (vicepresidente de infraestructura)	Sam Ruby
V.P., **Legal Affairs** (vicepresidente de asuntos jurídicos)	Sam Ruby
V.P., **Marketing and Publicity** (vicepresidente de marketing y publicidad)	Sally Khudairi
V.P., **Travel Assistance** (vicepresidente de asistencia de viajes)	Gavin McDonald
V.P., **W3C Relations** (vicepresidente de relaciones con W3C)	Andy Seaborne

4.4.1 Registros públicos

La Apache Software Foundation, como una cuestión de rutina, ofrece a la comunidad diversos registros públicos relativos a sus operaciones. Aquí hay una serie de documentos sobre las operaciones corporativas de la Fundación:

- El Certificado de incorporación de la Apache Software Foundation.
- El Certificado para la renovación de la Apache Software Foundation.
- El consentimiento por escrito del incorporador de The Apache Software Foundation, en lugar de reunión organizativa.
- Los Estatutos de la Fundación Apache Software.
- El calendario y actas.

4.5 Project Management Committees (PMC)

Los Comités de Gestión de Proyectos, se establecen por acuerdo de la Board of Directors (la junta), que será responsable de la gestión activa de una o más comunidades, que también se identifican por resolución de la Junta.

Cada PMC consiste en por lo menos un officer de la ASF, que será designado presidente, y puede incluir uno o más miembros de la ASF.

El presidente de la PMC es nombrado por la Junta Directiva y es un officer de la ASF (Vicepresidente, en la actualidad son 130 VP). El presidente tiene la responsabilidad principal de la Junta, y tiene la facultad de establecer normas y procedimientos para la gestión del día a día de las comunidades para las que el PMC es responsable, incluyendo la composición de la propia PMC.

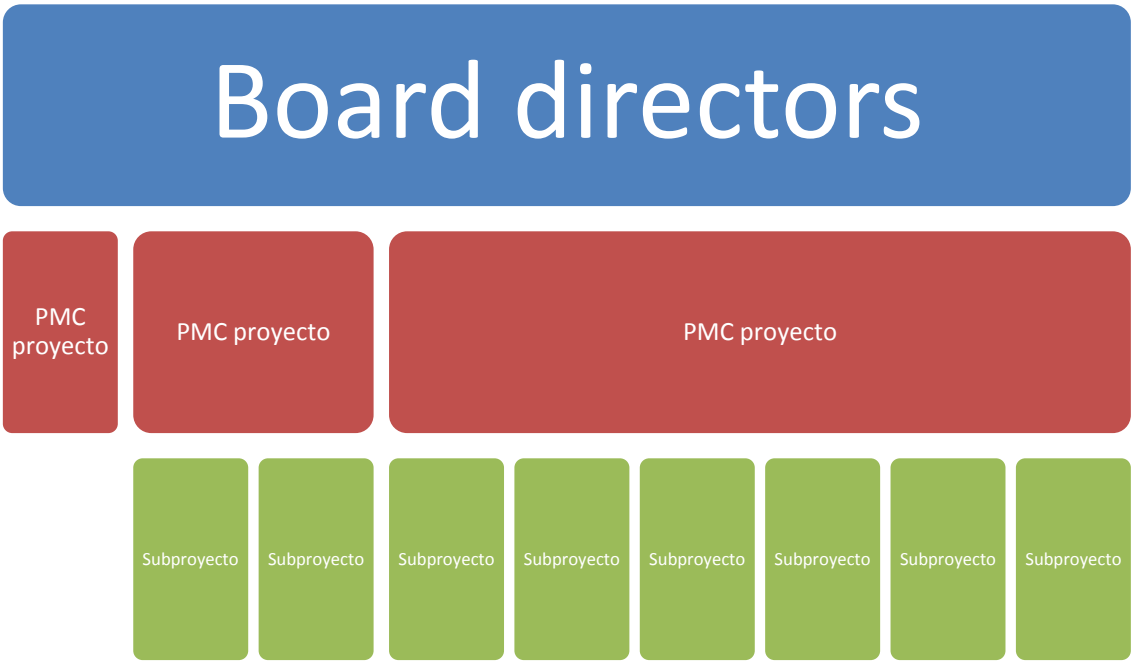

Figura 4. Niveles de proyectos y sub-proyectos asignados a un PMC

El papel de la PMC desde una perspectiva Fundación es la supervisión.

El papel principal de la PMC no es desarrollar código, sino garantizar que se aborden todas las cuestiones legales, que procedimiento se sigue, y que todos y cada lanzamiento es el producto de la comunidad en su conjunto. Esa es la clave de nuestros mecanismos de protección legal.

En segundo lugar el papel de la PMC es promover el desarrollo a largo plazo y la salud de la comunidad en su conjunto, y para garantizar una amplia supervisión equilibrada y con colaboración por una comunidad que gira en torno a unos pocos individuos que están trabajando prácticamente sin oposición.

La Junta tiene la facultad de poner fin a un PMC en cualquier momento por resolución.

4.6 Roles

La meritocracia permite diversas funciones:

Figura 5. Roles de la meritocracia en la ASF

4.6.1 Usuario

Un usuario es una persona que utiliza un software. Su contribución a los proyectos Apache, radica en proporcionar información a los desarrolladores en forma de informes de errores y sugerencias de características. También los usuarios que participan en la comunidad Apache, ayudando a otros usuarios en las listas de correo y foros de soporte al usuario.

4.6.2 Desarrollador

Un desarrollador es un usuario que contribuye a un proyecto en forma de código o en la documentación. Se toman medidas adicionales para participar en un proyecto, son activos en la lista de correo de desarrolladores, participan en las discusiones, proporcionan parches, documentación, sugerencias y críticas. Los desarrolladores también son conocidos como contribuyentes.

4.6.3 Committer

Un committer es un desarrollador que se le dio acceso de escritura al repositorio de código y tiene un firmado un Acuerdo de licencia Colaborador (CLA) en el archivo. Tienen una dirección de correo apache.org. No tiene que depender de otras personas para los parches, y en realidad está tomando decisiones a corto plazo para el proyecto.

El PMC puede (aunque tácitamente) acordar y aprobar en permanencia, o pueden rechazarlo.

4.6.4 Miembro PMC (Comité de Gestión de Proyecto)

Un miembro de PMC (Comité de Gestión de Proyecto) es un desarrollador o un committer que fue elegido debido al mérito de la evolución del proyecto y la demostración de compromiso. Ellos tienen acceso de escritura al repositorio de código, una dirección de correo apache.org, el derecho a votar a favor de las decisiones relacionadas con la comunidad y el derecho de proponer un usuario activo de committership. El PMC en su conjunto es la entidad que controla el proyecto, nadie más.

4.6.5 Presidente de PMC (Comité de Gestión de Proyecto)

El Presidente de un Comité de Gestión del Proyecto (PMC), es nombrado por el Board Directors de uno de los miembros del PMC. El PMC en su conjunto, es la entidad que controla y dirige el proyecto. El Presidente es la interfaz entre el Board directors y el proyecto.

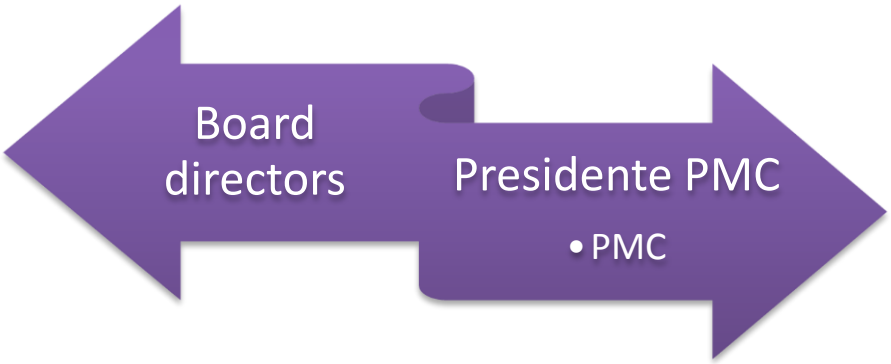

Figura 6. Flujo de gestión entre la Board directors, presidente de la PMC y la PMC

4.6.6 Miembro de la ASF

El ser miembro en The Apache Software Foundation, es un privilegio y es sólo por invitación. Un miembro es una persona que fue nominada por los miembros actuales y es eligió debido al mérito de la evolución y el progreso de la fundación. Esto se demuestra por lo general a través de su eficacia al contribuir con el desarrollo de software de código abierto en uno o más de los proyectos de la Fundación. Legalmente, un miembro es un "shareholder" de la fundación, uno de los propietarios. Ellos tienen el derecho de elegir a la Board of Directors, también a presentarse como candidato a las elecciones de la Board (junta) y de proponer un miembro para el Committer. Además tienen el derecho de proponer un nuevo proyecto para la incubación (veremos más adelante lo que esto significa). Los miembros coordinan sus actividades a través de su lista de correo y a través de una reunión anual.

Actualmente la ASF está constituido por 422 miembros aproximadamente, más 55 miembros eméritos y más 3 miembros fallecidos. Por lo tanto, la ASF se rige por la comunidad a la que sirve más directamente a la gente que colaboran en sus proyectos.

4.7 Patrocinadores

La Fundación Apache Software no podría existir sin el continuo apoyo generoso de la comunidad. El Programa de Patrocinio ASF es la vía oficial para las contribuciones monetarias sustanciales, no dirigidas a la ASF. Es el método más cercano y directo para una empresa o persona para apoyar la ASF. A cambio, ASF hace reconocimiento oficial y agradece por la donación a través de la participación de PR (según corresponda de acuerdo al nivel de patrocinio), logo del patrocinador y un enlace en la página de agradecimientos de la ASF (logo y el enlace con sujeción a la aprobación ASF, enlaces incluyen 'rel = nofollow' a nivel de Bronce) y un logo oficial ASF para colocar en el sitio del Patrocinador, además de hacer comunicados de prensa en conjunto.

Hay cuatro niveles de patrocinio, que se define por la donación y los beneficios de retorno de patrocinio. No más del 25% del valor puede estar basada en servicios en especie.

Tabla 3. Clasificación de patrocinadores de la ASF.

Patrocinador	Donación a partir de	Patrocinadores
Platinum	100 mil	Citrix, Yahoo, Microsoft, Google, Facebook, Matt Mullenweg,
Gold	40 mil	AMD, Cloudera, Comcast, HP, Hortonworks, IBM,
Silver	20 mil	Basis Technology, Budget Direct, God Daddy, Huawei, InMotion Hosting, Pivotal, PSW Group, WANdisco
Bronze	5 mil	Accor, BlueNog, Cloudsoft Corporation, Digital Primates, FuseSource, Intuit, Liip AG SA Ltd, Lucid Imagination, Object Engineering GmbH, Talend, Twitter, Two Sigma Investments.

Figura 7. Logotipos de los patrocinadores

Las donaciones hechas al programa de patrocinio deben coordinarse con las relaciones públicas de la ASF y el equipo de recaudación de fondos (fundraising at apache dot org).

También se encuentra organizaciones que proporcionan apoyo a la infraestructura como:

- Oregon State University Open Source Lab (OSUOSL) (EE.UU.), SURFnet y FUB (UE), que proporcionan Servidor de hosting y ancho de banda.
- Sun, IBM, HP, Dell y Calxeda, con donación de servidores.
- VMware, con donación de software de virtualización.
- Thawte, con donación de los certificados SSL comodín.

4.8 Como unirse a la ASF

La Apache Software Foundation es una meritocracia, lo que significa que para llegar a ser un miembro primero debe ser contribuir activamente a uno o más de los proyectos de colaboración de la Fundación. Nuevos candidatos a miembros son nominados por un miembro existente y luego sometidas a votación, una mayoría de los miembros existentes deben aprobar a un candidato con el fin de que el candidato para ser aceptada.

4.9 Filosofía

Si bien no hay una lista oficial, estos seis principios han sido citados como las creencias fundamentales de la filosofía detrás de la fundación, que normalmente se conoce como "The Apache Way":

- Desarrollo colaborativo de software
- La licencia comercial estándar de uso
- Software de alta calidad constante
- Interacción respetuosa, honesta
- Fiel aplicación de las normas
- De seguridad como una característica obligatoria

Todos los proyectos de ASF comparten estos principios.

4.10 Comunicación

La comunicación se realiza a través de listas de correo. Estos identifican “salas de reuniones virtuales” donde las conversaciones se producen de forma asincrónica, lo que permite crear archivos que son consultados por las diversas comunidades. Esta comunicación es un requisito general para todos los grupos, ya que están distribuidos geográficamente y así cubrir todas las zonas en distinto horario.

Algunos proyectos, además, utilizan la mensajería más sincrónico (por ejemplo, IRC o mensajería instantánea). La comunicación de voz es poco frecuente, por la barrera del idioma (hablar es más difícil de entender que el texto escrito).

Sin embargo a veces las listas de correo internas deben ser privadas. Nunca se debe divulgar dicha información en público sin la autorización expresa de la lista.

4.11 Documentación

Cada proyecto es responsable de su propia página web del proyecto. Más información para ayudar committers, desarrolladores y PMC está disponible en ASF Infraestructura.

4.12 Infraestructura

La ASF no tiene oficinas o edificios, es una entidad intangible que sólo existe en el Internet. Su única existencia física es la infraestructura técnica que le permite funcionar.

El equipo de infraestructura es responsable de la gestión y administración del sistema día a día y el funcionamiento de los distintos hardware que ejecuta los servicios mencionados. Ellos también son responsables de la aprobación de la instalación de un nuevo sistema o software en las máquinas ASF.

El equipo Infraestructura ASF proporciona y administra toda la infraestructura y los servicios de la Fundación Apache Software, y para cada proyecto en la Fundación. Esta infraestructura incluye:

- Las diferentes máquinas y sus sistemas operativos
- Las listas de correo
- Los sistemas de control de versiones
- Las cuentas del committer
- El sistema de espejos de distribución
- Una variedad de sistemas de seguimiento de problemas

El equipo Infraestructura ASF es responsable de la administración de sistemas, y así apoyar a los proyectos existentes en la Fundación Apache Software, el equipo de Infraestructura ASF trabaja duro para proveer servicios y la configuración para cada nuevo proyecto que se une a la Fundación a través de la incubadora de Apache.

El equipo Infraestructura ASF se compone de committers ASF, algunos de los cuales se pagan al ser contratados como sysadmin. Al igual que todos los proyectos de ASF, los committers no contratados trabajan como voluntarios.

La toma de decisiones funciona un poco diferente para el equipo de infraestructura ASF de lo que podría ser utilizado en promedio con los proyectos de desarrollo de software de la ASF. A veces se vota, o la gente hace cosas porque tiene sentido hacerlas, y es que las personas están familiarizadas con las configuraciones de las máquinas y sus sistemas, además de conocer el funcionamiento del equipo de la infraestructura.

A continuación se presenta una lista de servicios prestados por la infraestructura:

Detalles de la infraestructura

- Twitter stream
- Blog
- Como ser voluntario de la infraestructura
- Como informar de problemas y solicitar recursos de un proyecto
- Infraestructura de listas de correo
- Como editar el sitio web de infraestructura

Máquinas y servicios

- Lista de máquinas (http://www.apache.org/dev/machines.html) soportadas en:
    - FreeBSD, SunOS, Ubuntu, MacOSX, VCenter, Virtual Machines, Zones, Jails-Services, Jails-TPLs,
    - SSH RSAKeys, SSH ECDSA Keys, SSL Keys
- Lista de servicios:
    - sitios web:
        - Sitios web de ASF: projects.apache.org, www.apache.org, people.apache.org.
        - Cuentas de Shell
        - Espacio de web personal para commiter
    - e-mail
    - seguimiento de problemas y sus repositorios de integración
        - Bugzilla
        - Jira
        - Directrices para informar errores
    - repositorios de código
        - Repositorio de Subversion (SVN), para controlar versiones de infraestructura
        - ViewVC (interfaz del navegador para el repositorio principal)
        - Guía para committers y otros miembros de la comunidad
        - Solo lectura de espejos Git de código base SVN
        - Repositorio experimental de Git grabables
    - Servidores virutales
    - Distribuciones
        - Directorio de distribuciones de la ASF
        - Archivo histórico de distribuciones
        - rsync
        - Página de status de los espejos públicos
        - Nexus, Archiva, Pear
    - Wikis, Blogs, DNS, IRC services
- Host público y estado del servicio

Servicios internos

- Usamos fork of CKL (CloudKick Logger) el objetivo es proporcionar un sistema de registro voluntario para tareas de administración Unix realizadas por personas con privilegios de root.

La infraestructura está desplegada en la producción de una serie de softwares Apache, incluyendo: HTTP Server (httpd), Subversion, SpamAssassin, Tomcat, Traffic Server, mod_perl, RAT (incubating), Gump and Continuum, Steve, mod_mbox, Pulse and Orthrus, Solr, Directory Studio, una selección de proyectos internos (incluyendo el índice de committers y committers URL Shortener), and the Apache CMS.

4.13 Toma de decisiones

Los proyectos son normalmente auto gobierno y conducido por las personas que se ofrecen voluntariamente para trabajar en él. Esto se refiere a veces como "Hacer-ocracia", es una estructura organizativa en la que los individuos eligen las funciones y tareas por sí mismos y las ejecutan; porque pueden hacerlo.

Cuando se requiere la coordinación, las decisiones se toman con un enfoque de consenso perezoso: algunos votos positivos con ningún voto en contra es suficiente para ponerse en marcha.

La votación se realiza con los números: 1 es un voto positivo, 0 es abstenerse, no tiene opinión, -1 es un voto negativo.

Figura 8. Votación para la toma de decisiones

Las reglas exigen que si hay un voto negativo, se incluya una propuesta alternativa o una explicación detallada de las razones del voto negativo.

La comunidad trata de reunir un consenso sobre una propuesta alternativa que resuelve el problema. En la gran mayoría de los casos, las preocupaciones principales del voto negativo se pueden abordar, formando consenso. Este es un proceso importante considerando que es una indicación de una comunidad saludable.

4.14 Operación

Todos los proyectos se componen de voluntarios y nadie (ni siquiera los miembros o funcionarios) son pagados directamente por las bases de su trabajo. Hay muchos ejemplos de committers que se pagan para trabajar en los proyectos, pero nunca por los propios cimientos, sino más bien por las empresas o instituciones que utilizan el software y quiere mejorarlo o mantenerlo.

4.15 Licencia Apache 2.0 o ASL (Apache Software Licencia 2.0)

La licencia APACHE 2.0 fue creada por The Apache Software Foundation (ASF), como ya se ha mencionado es una organización sin ánimo de lucro cuya finalidad es dar soporte al desarrollo de proyectos de software de open source, desde una perspectiva de hardware, de negocio y legal.

La Apache Software Foundation (ASF) tiene una gran importancia en el ámbito del software de open source, por lo que las licencias generadas por APACHE han sido utilizadas en una multiplicidad de proyectos.

Existen tres versiones de la licencia: ASL 1.0, ASL 1.1 y la ASL 2.0. Desde enero de 2004, todo el software de la Apache Software Foundation (ASF) se publica bajo la licencia ASL 2.0.

La licencia APACHE 2.0 ha sido certificada por la Open Source Initiative como licencia de open source.

4.15.1 Rasgos distintivos

La licencia ASL. 2.0 se caracteriza por:

- Ser una licencia permisiva: permite realizar todo tipo de operaciones con el código fuente y desarrollos propietarios.
- No incluye una cláusula copyleft.
- Sus destinatarios son desarrolladores y usuarios, pudiendo ser individuos, empresas y administraciones públicas.
- Establece un régimen de exoneración total de responsabilidad y ausencia de garantía sobre el software licenciado.
- Permite que el Licenciante establezca, de mutuo propio, un régimen de responsabilidad respecto al software, incluida la incorporación de una tarifa.
- Otorga paralelamente una licencia de uso de patente sobre el software, la que será revocada automáticamente si el Licenciatario inicia acciones legales por infracción de patentes contra el Licenciante.

4.15.2 Obligaciones de la licencia

La licencia APACHE 2.0 incluye una serie de obligaciones:

- Incluir siempre una copia de la licencia.
- Indicar cualquier modificación.
- Mantener los avisos de titularidad (copyright notice), marca o patentes.
- Inclusión de fichero notice.txt con indicaciones de autoría, modificaciones y otros contenidos de carácter legal.
- Se prohíbe utilizar el nombre del autor para publicar el software.
- Se prohíbe el uso de la marca del Licenciante.

4.15.3 Donaciones de software

Cuando un individuo o corporación decide donar una cantidad de software o la documentación existente a uno de los proyectos de Apache, se necesita ejecutar un Convenio oficial de Donación de Software (SGA, Software Gran Agreement), con la ASF. Normalmente, esto se hace después de la aprobación de la Incubadora de la ASF o una de los PMC´s, ya que la ASF no aceptará software a menos que haya una comunidad viable para apoyar un proyecto de colaboración.

4.16 Estatutos de la ASF

Los estatutos de The Apache Foundation, se componen de nueve artículos con sus secciones respectivamente, los cuales regulan las relaciones entre los diversos actores y los miembros de la fundación.

Artículo I. Oficinas Comerciales: localizada en Delawere EEUU, pero puede estar en cualquier parte del mundo, según interés de la fundación.

Artículo II. Domicilio Social y Agentes Registrados: Establecimiento del certificado de constitución del domicilio y agentes registrados. Además de tratar otros posibles estados.

Artículo III. Reuniones de los Miembros: Establecimiento de lugar de encuentro de la reunión anual, reunión extraordinaria, y notificación que dará lugar no menos de 10 días y no más de 60 días. Además, de notificaciones de reuniones aplazadas, renuncias de avisos, fijación de la fecha de registro, registro de miembros con derecho a voto, miembro de quórum, votaciones, proxies, acción de los miembros sin una reunión.

Artículo IV. Miembros: este artículo describe el proceso de admisión de miembros, el nombramiento de miembros eméritos, la conversión voluntaria de afiliación para ser miembro emérito, el restablecimiento como miembro pleno un miembro emérito, así como el retiro voluntario de los miembros, la terminación de la membresía por votación de dos tercios de los miembros de la corporación, así como el efecto del retiro o terminación.

Artículo V. Directores: este artículo describe los poderes, la cualificación, compensaciones, el número de directores (inicialmente 9), elección y periodo, la dimisión y cese de los directores, vacantes, quórum y votación, lugar de reuniones, notificación y tiempo para la convocatoria de reuniones, resolver conflictos de interés.

Artículo VI. Oficiales: se describen los deberes del presidente de la junta, vicepresidente, secretario, subsecretario, tesorero, tesorero adjunto. Además de tener la facultad de establecer los PMC a través de elección y su período, también pueden destituir a los miembros de la PMC y nombrar nuevos miembros.

Artículo VII. Libros y Registros: Los libros, registros y minutas pueden estar en forma escrita o en cualquier otra forma susceptible de ser convertida en forma escrita claramente legible en un plazo razonable. Además en este artículo se trata la inspección de los derechos de los miembros.

Artículo VIII. Estado sin fin de lucro: esté artículo manifiesta que es una sociedad organizada y que se manejara como una corporación de miembros sin fines de lucro.

Artículo IX. Sello corporativo: trata de un sello con el nombre de la empresa, que puede ser un fasimil, un grabado, un impreso, o un sello de impresión.

Artículo X. Enmienda: Estos estatutos pueden ser alterados, modificados o derogados por el Consejo de Administración o por los miembros, y los nuevos Estatutos podrán ser adoptadas por el Consejo de Administración o por los miembros.

Artículo XI. Límites de responsabilidades de la Board of Directors.

Artículo XII. Indemnización de oficiales y directores.

Artículo XIII. Disposiciones generales: está relacionado con las finanzas de la fundación, como expedición de cheques, ejercicio fiscal, no hay prestamos, depósitos, contratos, ejecución de contrapartes.

Estos estatutos fueron leídas, aprobadas y adoptadas POR EL CONSEJO DE ADMINISTRACIÓN DE LA APACHE SOFTWARE FOUNDATION EN LA 1 ª JORNADA DE JUNIO DE 1999.

(Firmado) Jim Jagielski, Secretario

4.17 Incubadora

El proyecto Incubadora, es el camino de entrada a la Apache Software Foundation (ASF), para proyectos y bases de código que deseen ser parte de los esfuerzos de la Fundación. Todas las donaciones de código de organizaciones externas y proyectos externos existentes que deseen unirse a Apache entran a través de la Incubadora.

El Apache Incubadora tiene dos objetivos principales:

- Asegúrese de que todas las donaciones son de acuerdo con las normas legales de la ASF.
- Desarrollar nuevas comunidades que se adhieren a los principios rectores de la ASF.

La incubadora es responsable de:

- Filtrado de las propuestas sobre la creación de un nuevo proyecto o subproyecto.
- Ayuda a la creación del proyecto y la infraestructura que se necesita para operar.
- Supervisa y es mentor de la comunidad que se incubaron con el fin de que logren alcanzar un entorno meritocrático abierto.
- Evaluar la madurez del proyecto incubado, ya sea promoviendo a proyecto oficial de estad / subproyecto o retirarlo, en caso de fallo.

Debe tenerse en cuenta que la incubadora (al igual que la junta) no realiza el filtrado sobre la base de las cuestiones técnicas. Esto se debe a la fundación respeta y sugiere variedad de enfoques técnicos. No teme la innovación o la confrontación interna, incluso entre los proyectos que se solapan en la funcionalidad.

La incubadora filtra proyectos sobre la base de la probabilidad de que se conviertan en comunidades meritocráticas exitosas.

Los requisitos básicos para la incubación son:

- Una base de código de trabajo inicial, con la finalidad de reconocer su mérito.
- La intención de donar los derechos de autor del software y la propiedad intelectual que puede contener a la base. Esto permite un derecho irrevocable, redistribución y trabajar en el código, sin temer el lock-in, para la comunidad.
- Un miembro patrocinador ASF o funcionario, esta persona actuará como el principal mentor, dando indicaciones al proyecto, ayudando en los detalles del día a día y mantener el contacto con el PMC de la incubadora.

El período de incubación sirve normalmente para estimar si:

- El proyecto es capaz de aumentar la diversidad de su base committer y jugar con las reglas meritocráticas de la fundación.
- Puede parecer algo fácil de lograr, pero hay que recordar que en un entorno voluntario y altamente selectivo, atraer nuevos committers no es automático.

La diversidad de committership es importante por dos razones principales:

- Da la estabilidad a largo plazo para el desarrollo del proyecto: con todos los desarrolladores afiliados a la misma entidad, con un vínculo creado.
- Se da una mayor variedad de visiones técnicas: algo que garantiza una mejor adherencia con el medio ambiente y las necesidades del usuario, por lo tanto un cambio mayor de encontrar uso en la vida real del software.

Roles.

El Comité de Gestión del Proyecto Incubadora (IPMC) supervisa los procesos de incubación. El PMC acepta proyectos y proporciona el apoyo técnico y administrativo necesario, se revisa; regularmente los proyectos incubados, los propone para la terminación, continuación o escalamiento.

Presidente de la Incubadora de PMC, es la persona designada por la junta y tiene la responsabilidad de supervisar los proyectos de la Incubadora, sus políticas y la política de implementación.

El candidato, es un proyecto que se propone para la incubación. La aceptación de un candidato estará sujeto al método de votación formal del Patrocinador. Si se acepta

el voto, el patrocinador solicitará a la Incubadora PMC aceptar al candidato como un Podling bajo la incubación. El Patrocinador le asignará un mentor, que concederá la adhesión a la Incubadora PMC durante el proceso de incubación.

El campeón asiste al candidato en su presentación inicial a un patrocinador. Ayuda a educar al candidato sobre la manera de la ASF y preparar el proyecto de preguntas y cuestiones que puedan ser planteadas por la comunidad. También durante la incubación coordina la creación y entrega oportuna de informes a la junta podling´s. Solicita nuevos mentores si lo cree necesario.

El patrocinador es la entidad ASF que defiende el proyecto candidato como un digno colaborador y se compromete a supervisar al candidato en cuestión. Tras la aceptación de la Incubadora de PMC´s, el candidato se convierte en un Podling (un proyecto en la fase de revisión, antes de la graduación) bajo su cuidado.

El mentor es un miembro de ASF permanente que tiene responsabilidades específicas hacia la Incubadora PMC, el patrocinador y los miembros del proyecto Podling asignado.

Por ejemplo, el mentor ayuda, guía y protege el proyecto sobre la base de su experiencia con el proceso y la familiaridad con las políticas y procedimientos de incubación. Un mentor mantiene la comunicación oportuna entre el PMC y los miembros Podling (la comunidad) sobre las decisiones que afectan al proyecto, en otras palabras, el mentor es el punto de contacto entre la comunidad y la incubadora de PMC.

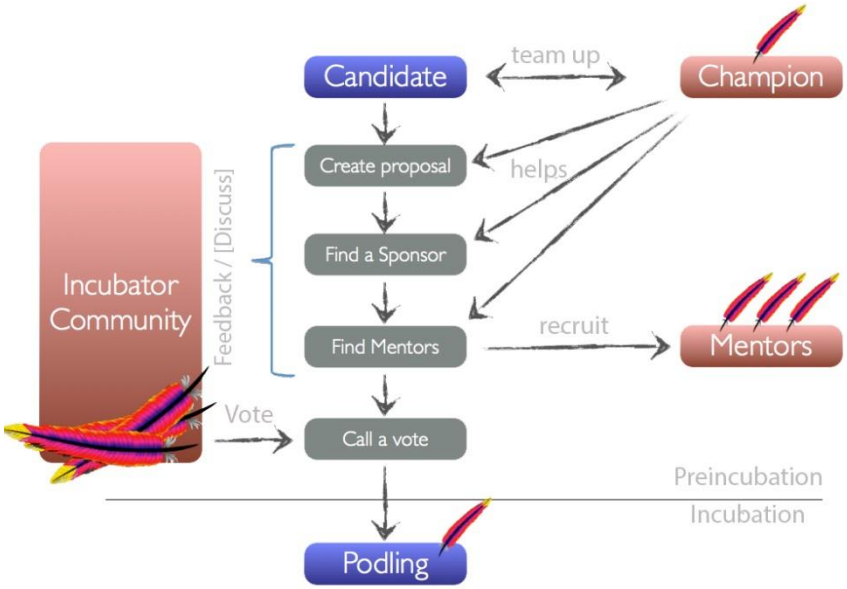

Figura 9. Diagrama de flujo

Las fases del proceso. El proceso se muestra en figura 10 tiene tres fases principales: establishment, acceptance, y review (http://incubator.apache.org/incubation/Process_Description.html).

La fase de Stablishment, consiste en encontrar a un champion, preparando una propuesta, y su presentación. Entonces, el champion asigna el status de candidato al proyecto.

En la fase de Acceptance, el patrocinador evalúa la propuesta y la acepta o la rechaza. Si es aceptado, el patrocinador propone la Incubadora PMC la escalada del proyecto candidato a status podling, y el patrocinador asigna a un mentor para el proyecto.

Durante la fase de **Review** iterativa, la Incubadora PMC evalúa el status del proyecto.

Tres resultados son factibles:

- El proyecto puede ser terminado.
- El proyecto puede ser continuado (durante el cual el proyecto debe adoptar las recomendaciones de desarrollo), o;
- El proyecto puede ser contratado (el proyecto es aceptado en la ASF, ya sea como un sub-proyecto o un proyecto de nivel superior).

El proyecto puede ofrecer versiones mientras está en la fase de revisión, pero la ASF no apoyará los lanzamientos porque no ha aceptado plenamente el proyecto Podling como parte de la fundación.

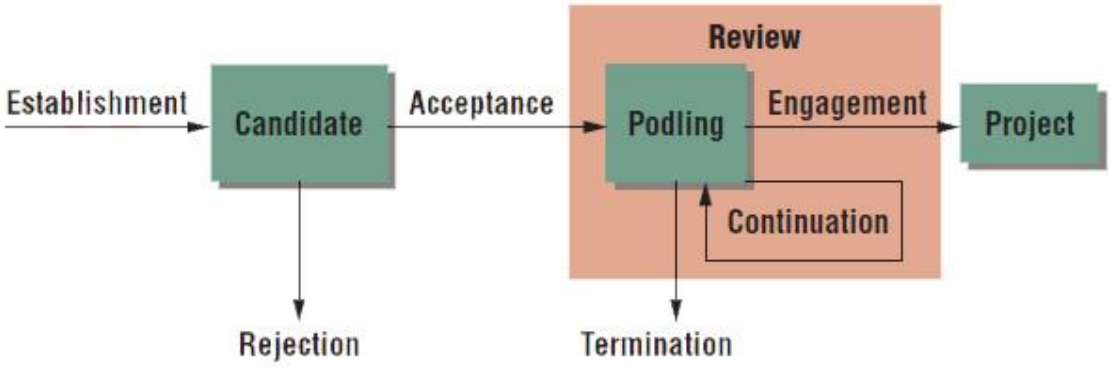

Figura 10. Proceso de incubación las tres fases establecidas por Apache

4.17.1 Análisis cuantitativo de Incubadora Apache

La Universidad Politécnica de Madrid ha presntado un artículo llamado Apache and Eclipse: Comparing Open Source Project Incubators", donde se han identificado los principios de cada comunidad, los roles y las fases, también se estudió el comportamiento de la incubadora mediante la aplicación de análisis estadístico sencillo de los datos disponibles. Y ha sido tomado como base para hacer este análisis de la

situación actual al 2013 de los proyectos incubados, graduados y retirados de la incubadora.

Desde la creación de la Incubadora, la ASF ha incubado 190 proyectos, de los cuales 122 se han graduado con éxito, 36 están a la espera de la graduación, y 32 se encuentran cerrados. Para evaluar el proceso de incubación, separamos los proyectos en graduados y los que todavía están en la incubadora.

Proyectos restantes en incubación. El lado izquierdo de la figura 11 muestra que después de 10 meses en incubación, 11 de los 36 proyectos (con más de cinco committers en 32 proyectos) logró una comunidad consolidada. Este período corresponde del año 2008 al 2013 de la incubadora y por lo tanto a un mayor nivel de experiencia entre los mentores y patrocinadores que fueron responsables de la selección y apoyo a los proyectos candidatos. Por otro lado, de tres proyectos de incubación reciente, ha comenzado con más de 4 committers y otros tres con más de 25 committers, estos son proyectos de gran interés para los committers. Pero esto indica un posible riesgo potencial de contraer inactivos.

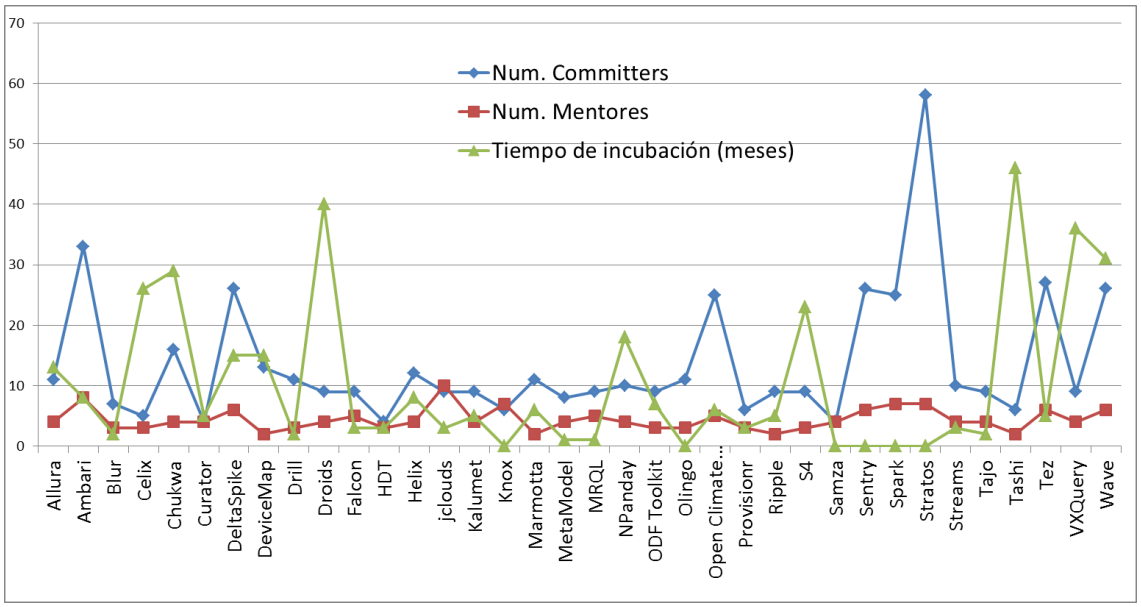

Figura 11. Proyectos restantes en incuabación

Proyectos Graduados. La figura 12 se representa los proyectos graduados de la incubadora desde su creación desde el primer año se puede observar que tuvo un proyecto graduado casi inmediatamente de entrar a la incubadora, esto se debe a que se desarrolló a la par con el proyecto de incubación. Podemos observar que 50 de los 122 proyectos se graduaron en el periodo 2007 y 2012, esto quiere decir que cada 5 años se está teniendo un gran número de proyectos graduados. Y como podemos observar el incremento de los proyectos se duplica cada 5 años.

En la figura 13. Observamos los proyectos graduados y su relación con el número de committers y los meses que han estado dentro de la incubadora, hasta su

graduación. Y podemos ver que 23 de los 122 proyectos incubados, están por encima de los 30 meses de incubación.

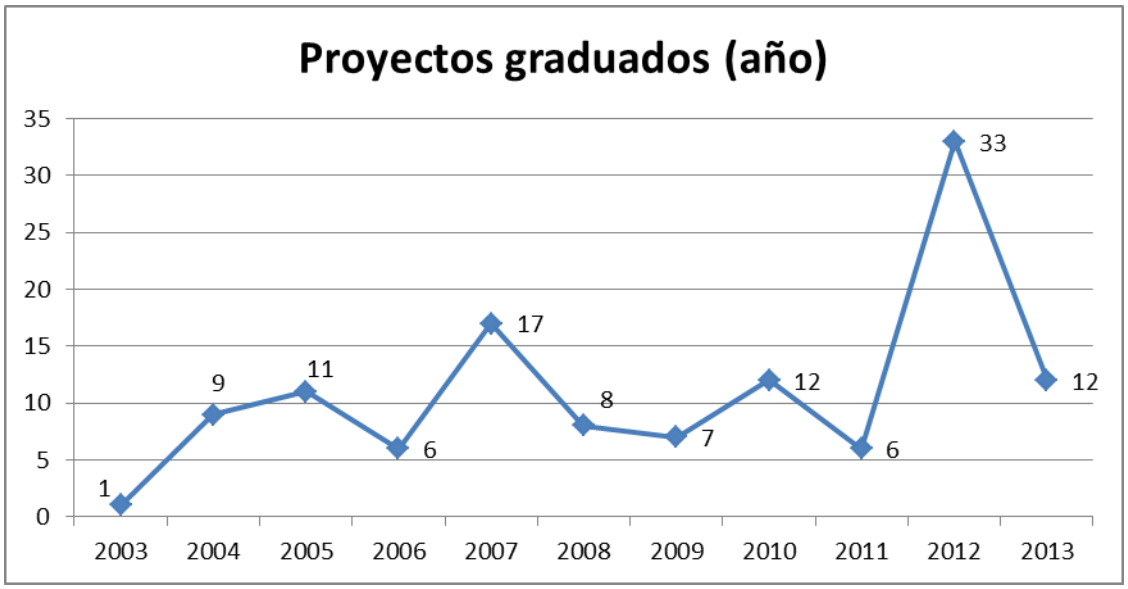

Figura 12. Proyectos graduados de la incubadora

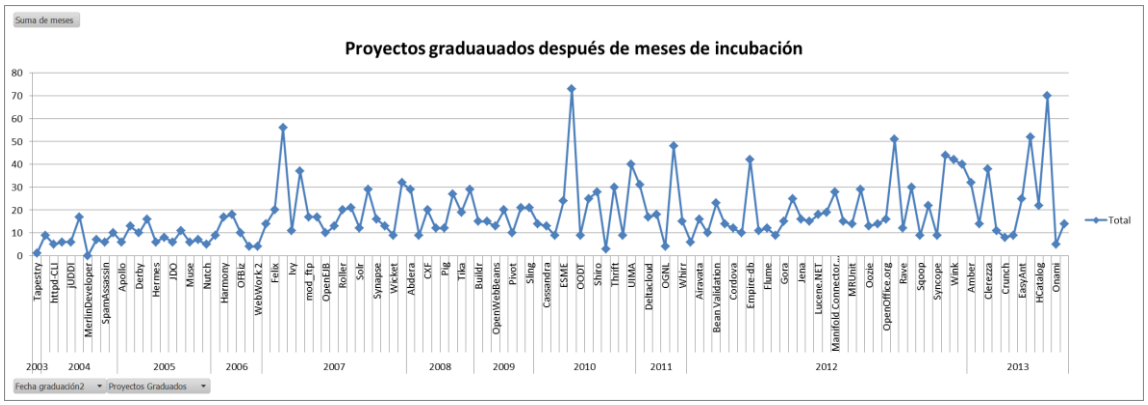

Figura 13. Proyectos graduados de la incubadora y los meses transcurridos

Estimar o definir la duración de la incubación razonable es difícil, porque la comunidad que apoya estos proyectos se compone de desarrolladores voluntarios. Sin embargo, sobre la base de los datos observadas, se considera que un buen patrón de estimación del tiempo de incubación para los subproyectos es de 8 meses, más o menos un mes.

Para los proyectos de alto nivel, un tiempo de incubación factible podría ser de 18 meses, más o menos un mes. Sin embargo, los niveles de actividad durante la incubación son tan importantes como la propia duración de sí mismo. En este sentido, largos periodos de inactividad no sólo aumentan el tiempo de incubación, pero también pueden indicar un riesgo mayor de fracaso del proyecto.

En resumen, las variables más importantes que afectan el tiempo de incubación de impacto y aumentan los riesgos del proyecto son

- El período periodo de inactividad del proyecto,
- El compromiso de nuevos committers para lograr una masa crítica, y la entrega de una liberación anticipada con suficiente impacto técnico.

4.18 Ciclo de vida de un proyecto de la ASF

4.19 Tamaño de la ASF

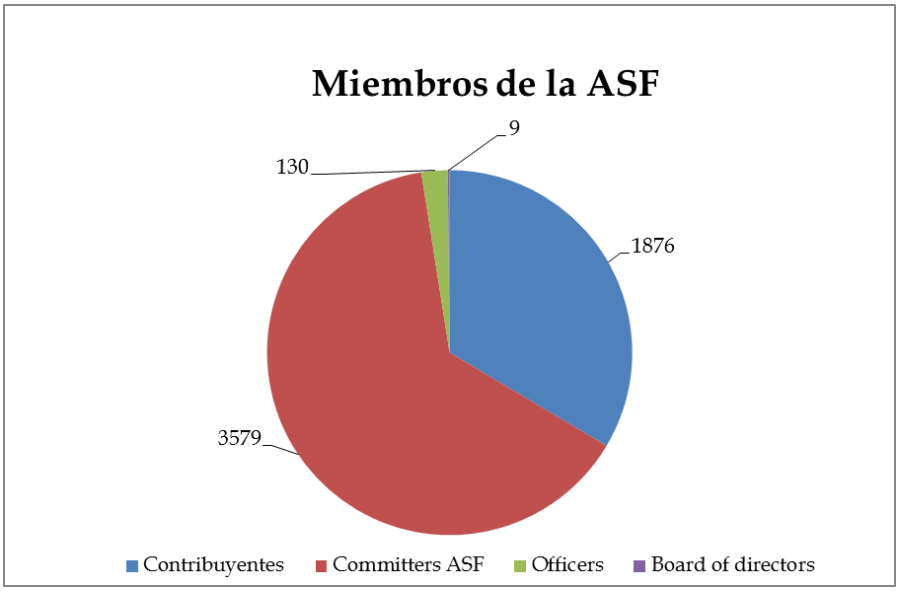

Figura 14. Número de miembros de la ASF.

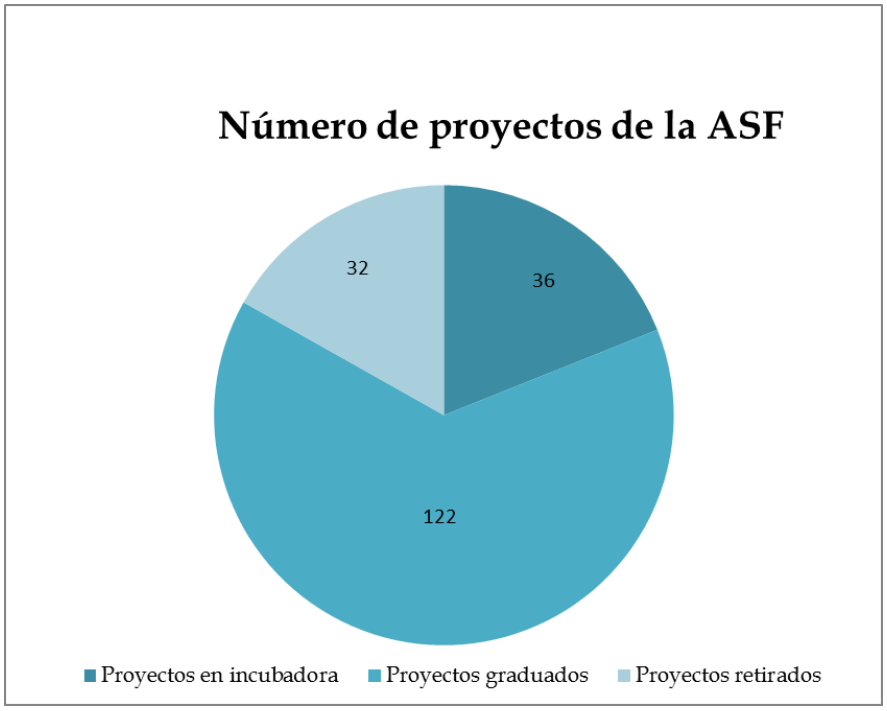

Figura 15. Número de proyectos de la ASF.

Tabla 4. Lista de 36 proyectos actuales en la Incubadora.

Allura	Ambari	Blur	Celix	Chukwa	Curator
DeltaSpike	DeviceMap	Drill	Droids	Falcon	Hadoop Development Tools
Helix	jclouds	Kalumet	Knox	Marmotta	MetaModel
MRQL	NPanday	ODF Toolkit	Olingo	Open Climate Workbench	Provisionr
Ripple	S4	Samza	Sentry	Spark	Stratos
Streams	Tajo	Tashi	Tez	VXQuery	Wave

Tabla 5. Lista de 122 proyectos graduados.

Tapestry	MerlinDeveloper	jUDDI	JaxMe	Pluto	Geronimo
XMLBeans	SpamAssassin	Lenya	httpd-CLI	log4cxx	Directory
MyFaces	iBATIS	Nutch	Muse	Hermes	Apollo
Derby	Beehive	JDO	Jackrabbit	Tobago	WebWork 2
Harmony	OFBiz	Cayenne	Synapse	Solr	log4net
mod_ftp	ActiveMQ	Roller	Felix	Trinidad	OpenJPA
OpenEJB	Wicket	Ode	ServiceMix	Ivy	stdcxx
Woden	FtpServer	CXF	Hama	Tuscany	Pig
Tika	Abdera	CouchDB	Qpid	Buildr	Sling
Sanselan	PDFBox	Click	Pivot	OpenWebBeans	Shindig
Chemistry	Cassandra	Subversion	Log4php	UIMA	Traffic Server
Shiro	Thrift	OODT	ESME	Aries	River
Libcloud	Whirr	OGNL	Deltacloud	ACE	Gora
Empire-db	Bean Validation	OpenNLP	Lucy	Rave	Sqoop
Accumulo	RAT	Jena	Mesos	Manifold Connector Framework (ManifoldCF)	MRUnit
Giraph	VCL	Flume	Lucene.NET	DirectMemory	Oozie
Any23	SIS	Stanbol	Airavata	Bigtop	Isis
OpenOffice.org	Cordova	Kafka	Syncope	Wookie	Nuvem
Wink	Flex	Etch	Amber	Openmeetings	HCatalog
Clerezza	Crunch	EasyAnt	Bloodhound	CloudStack	cTAKES
Onami	JSPWiki				

Tabla 5. Lista de 32 proyectos retirados.

AltRMI	WSRP4J	Depot	Axion
JuiCE	Graffito	Agila	Lucene4c
TSIK	XMLBeans/C++	Heraldry	wadi
Kabuki	Yoko	Lokahi	XAP
NMaven	TripleSoup	RCF	Composer
Imperius	Bluesky	PhotArk	Olio
Kato	Stonehenge	SocialSite	HISE
Zeta Components	Alois	Kitty	AWF

Cada proyecto se asigna a una o más categorías (tabla 5) que intentan describir su propósito. Por lo que los proyectos pueden aparecer más de una vez si su nombre figura en varias categorías.

Tabla 5. Lista de 23 categorías a las que puede pertenecer a un proyecto.

network-server	big-data	build-management	cloud	content	database
ftp	graphics	http	httpd-module	javaee	library
mail	mobile	network-client	network-server	osgi	regexp
retired	testing	virtual-machine	web-framework	xml	

5 Eclipse

5.1 Historia

Líderes de la industria de Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft y WebGain formaron la Junta eclipse.org inicial de Comisarios en noviembre de 2001. A finales de 2003, este consorcio inicial había crecido a más de 80 miembros.

El Proyecto Eclipse fue creado originalmente por IBM en noviembre de 2001 y con el apoyo de un consorcio de proveedores de software. La Fundación Eclipse fue creado en enero de 2004 como una corporación sin fines de lucro independiente que actúa como administrador de la comunidad Eclipse. La organización sin fines de lucro independiente fue creada para permitir a un proveedor neutral y abierto de la comunidad, transparente que se establece en torno a Eclipse. Hoy en día, la comunidad Eclipse consiste de individuos y organizaciones de diversos sectores de la industria del software.

Toda la tecnología y el código fuente proporcionado para y desarrollados por esta comunidad de rápido crecimiento se pone a disposición libre de regalías a través de la Licencia Pública Eclipse.

Los desarrolladores estratégicos fundadores y Consumidores estratégicos eran Ericsson, HP, IBM, Intel, MontaVista Software, QNX, SAP y Serena Software.

5.2 La Fundación Eclipse

Eclipse es una comunidad de personas y organizaciones que desean colaborar en el uso comercial de software open source. Sus proyectos se centran en la construcción de una plataforma de desarrollo abierta formada por marcos extensibles, herramientas y tiempos de ejecución para la construcción, despliegue y gestión de software a través del ciclo de vida.

El propósito de Eclipse Foundation Inc., es avanzar en la creación, evolución, promoción y apoyo de la plataforma Eclipse y cultivar tanto una comunidad de código abierto y un ecosistema de productos complementarios, capacidades y servicios.

La Fundación Eclipse es financiada por las cuotas anuales de los miembros y se rige por un Consejo de Administración. Desarrolladores estratégicos y Consumidores estratégicos tienen asientos en el Consejo, al igual que los representantes elegidos por el complemento de proveedores y committers de código abierto. La Fundación emplea a tiempo completo equipo de profesionales para prestar servicios a la comunidad, pero no emplea los desarrolladores de código abierto, llamados committers, que en realidad trabajan en los proyectos de Eclipse. Committers Eclipse se emplean típicamente por organizaciones o son desarrolladores independientes que ofrecen voluntariamente su tiempo para trabajar en un proyecto de open source.

En general, la Fundación Eclipse ofrece cuatro servicios a la comunidad Eclipse:

- Infraestructura IT.
- Gestión de la IP.
- Proceso de desarrollo.
- Desarrollo de Ecosistemas.

Personal de tiempo completo están asociados con cada una de estas áreas y trabajar con la comunidad en Eclipse para ayudar a satisfacer las necesidades de las partes interesadas.

5.2.1 Infraestructura

La Fundación Eclipse administra la infraestructura de IT para la comunidad de open source Eclipse, incluyendo repositorios de código Git, bases de datos Bugzilla, listas de correo y foros orientados hacia el desarrollo, el sitio de descarga y el sitio web. La infraestructura está diseñada para brindar un servicio confiable y escalable para las committers desarrollo de la tecnología de Eclipse y los consumidores que utilizan la tecnología.

5.2.2 Propiedad Intelectual (IP)

Un aspecto importante de Eclipse es el objetivo de facilitar el uso de la tecnología de open source de los productos y servicios de software comerciales. Conscientemente promover y alentar a los proveedores de software a utilizar la tecnología Eclipse para la construcción de sus productos y servicios de software comerciales. Esto es posible por el hecho de que todos los proyectos de Eclipse están licenciados bajo la Licencia Pública Eclipse (EPL), licencia aprobada por la OSI.

La Fundación Eclipse también lleva a cabo una serie de medidas para tratar de evitar el pedigrí de la propiedad intelectual contenida en proyectos de Eclipse. El primer paso en el proceso es una debida diligencia de que está tratando de asegurarse de que todas las contribuciones son hechas por el propietario de los derechos que le corresponde y bajo la Licencia Pública Eclipse (EPL). Todos committers están obligados a firmar un acuerdo de committer que establece que la totalidad de sus contribuciones son un trabajo original y están siendo aportados en el marco del EPL.

Si un committer es patrocinado a trabajar en un proyecto de Eclipse por una organización miembro, entonces esa organización se le pide que firme un Acuerdo Committer miembros, para asegurar que los derechos de propiedad intelectual de la organización son aportados bajo el EPL.

El segundo paso es que el código fuente relacionado con todas las contribuciones que se desarrollan fuera del proceso de desarrollo de Eclipse se procesan a través de la Fundación Eclipse, dentro del proceso de aprobación de IP. Este proceso incluye el análisis de las contribuciones de código seleccionados para tratar de determinar la procedencia del código, y la compatibilidad de la licencia con el EPL. Las contribuciones que contienen código licenciado bajo licencias que no sean compatibles con el EPL, están destinados a ser examinados a través de este proceso de aprobación y por lo tanto no se agregan a un proyecto Eclipse. El resultado final es un nivel de

confianza de que la tecnología de liberación de proyectos Eclipse de código abierto, se pueden distribuir de manera segura en los productos comerciales.

5.2.3 Comunidad de desarrolladores

La comunidad Eclipse tiene una reputación bien ganada, por proporcionar software de calidad de una manera fiable y predecible. Esto se debe al compromiso de los committers y organizaciones que contribuyen a los proyectos de código abierto. La Fundación Eclipse también proporciona servicios y apoyo a los proyectos para ayudarles a alcanzar estos objetivos.

El personal de la Fundación ayuda a poner en práctica el proceso de desarrollo Eclipse. Este proceso ayuda a poner en marcha un nuevo proyecto y asegura que todos los proyectos de Eclipse se ejecutan de una manera abierta, transparente y meritocrático. Como parte de este proceso, la Fundación organiza opiniones de los miembros de la comunidad para garantizar una interacción constante entre los proyectos y los miembros en general.

La comunidad Eclipse organiza un tren de lanzamiento anual que proporciona una liberación coordinada de los proyectos de Eclipse que deseen participar. El tren de liberación que hace que sea más fácil para los consumidores intermedios adoptar las nuevas versiones de los proyectos ya que:

- Todos los proyectos están disponibles en el mismo horario, por lo que no tienen que esperar a los calendarios de proyectos independientes, y
- Un nivel de pruebas de integración que ocurre antes de la versión final para ayudar a identificar los problemas de proyectos cruzados.

5.2.4 Desarrollo de Ecosistemas

Un aspecto único de la comunidad Eclipse, y el papel de la Fundación Eclipse es la comercialización activa y la promoción de proyectos de Eclipse y más amplio ecosistema Eclipse. Un ecosistema saludable y vibrante que se extiende más allá de la comunidad de código abierto Eclipse para incluir cosas como productos comerciales basados en Eclipse, otros proyectos de código abierto utilizando Eclipse, la formación y los proveedores de servicios, revistas y portales de Internet, libros, etc, son todos clave para el éxito de la comunidad Eclipse.

Para ayudar en el desarrollo del ecosistema Eclipse, Eclipse Foundation organiza una serie de actividades, como eventos de marketing de cooperación con las empresas miembros, conferencias comunitarias, catálogos de recursos en línea (Eclipse Marketplace y el canal de Eclipse en YouTube), las reuniones bianuales de los Miembros y otros programas de promoción de toda la comunidad Eclipse.

5.2.5 Un modelo para el desarrollo Open Soruce

La Fundación Eclipse ha sido establecida para servir a los proyectos de código abierto Eclipse y la comunidad Eclipse. Como una organización sin fines de lucro independiente, la Fundación y el modelo de gobierno Eclipse garantiza que ninguna

entidad es capaz de controlar las estrategias, las políticas y operaciones de la comunidad Eclipse.

La Fundación se centra en crear un entorno para el éxito de proyectos de open source y para promover la adopción de la tecnología de Eclipse en soluciones comerciales y de open source. A través de servicios como IP Due Diligence, los trenes de liberación anuales, apoyo a la comunidad de desarrollo y el desarrollo del ecosistema, el modelo Eclipse de desarrollo de open source es un modelo único y probado para el desarrollo de open source.

5.3 Proceso de desarrollo de Eclipse

5.3.1 Principios

A continuación se describen los principios rectores utilizados en el desarrollo del proceso de desarrollo Eclipse.

5.3.1.1 Reglas de Compromiso para Open Source

Apertura: Eclipse está abierta a todos; Eclipse proporciona las mismas oportunidades para todos. Todo el mundo participa con las mismas reglas, no hay normas para excluir a ninguno de los colaboradores potenciales que incluyen, por supuesto, competidores directos en el mercado.

Transparente: Discusiones del proyecto, minutas, deliberaciones, planes de proyectos, planes de nuevas características y otros artefactos están abiertos, públicos y de fácil acceso.

Meritocracia: Eclipse es una meritocracia. Cuanto más contribuyas, más responsabilidad ganará. Los roles de liderazgo en Eclipse también está basado en el mérito.

5.3.1.2 Ecosistema Eclipse

Eclipse como una marca es la suma de sus partes (todos los proyectos), y los proyectos deberán procurar la mayor calidad posible en los marcos extensibles, herramientas ejemplares, procesos transparentes, y la apertura del proyecto.

La Fundación Eclipse tiene la responsabilidad de cultivar un ecosistema de productos complementarios, con capacidades y servicios. Por tanto, es un principio fundamental que el proceso de desarrollo de Eclipse garantice que los proyectos se gestionan en beneficio tanto de la comunidad de código abierto y los miembros del ecosistema. Para ello, se requiere que todos los proyectos de Eclipse, realicen las siguientes actividades:

- Comunicar sus planes de proyectos y los planes de nuevas características (mayor y menor) de una manera oportuna, abierta y transparente;
- Crear Frameworks de calidad de plataformas capaces de apoyar la creación de productos de calidad comercial en la parte superior de ellos, y

- Enviar extensibles, herramientas ejemplares que ayudan a activar una amplia comunidad de usuarios.

5.3.1.3 Tres Comunidades

Esencial para los fines de la Fundación Eclipse es el desarrollo de tres comunidades interrelacionadas alrededor de cada proyecto:

Colaboradores y Committers: una comunidad próspera, diversa y activa de los desarrolladores es el componente clave de cualquier proyecto Eclipse. Lo ideal sería que esta comunidad debe ser una comunidad abierta, transparente, inclusivo y diverso de Committers y (no Committer) colaboradores.

- Los proyectos deben tener varios objetivos para asegurar la diversidad de pensamiento y evitar depender de ninguna compañía u organización. Al mismo tiempo, reconocemos que la aplicación de una medida particular, la diversidad es una mala manera de alcanzar estos objetivos, sino que esperamos que el liderazgo de proyectos para ayudar a la diversidad evolucionar orgánicamente.
- La diversidad es un medio para un fin, no un fin en sí mismo, por lo que las metas de diversidad se diferencian por el proyecto sobre la base de los demás logros del proyecto (s).
- Los proyectos tienen que explicar sus esfuerzos de diversidad y logros durante críticas.

Usuarios: una comunidad de usuarios activa y comprometida es prueba positiva de que las herramientas de ejemplares del Proyecto son útiles y necesarias. Por otra parte, una gran comunidad de usuarios es uno de los factores clave en la creación de un ecosistema viable en torno a un proyecto de Eclipse, fomentando así de código abierto adicional y organizaciones comerciales a participar. Como todas las cosas buenas, una comunidad de usuarios requiere de tiempo y esfuerzo para llevar a buen puerto, pero una vez establecida es normalmente autosuficiente.

Adoptantes: son los activos comprometidos en adoptar plug-in de la comunidad de desarrolladores. Esta es la única manera de probar que un proyecto de Eclipse está proporcionando frameworks y herramientas extensibles accesibles a través de las API´s documentadas. La reutilización de los frameworks dentro de las empresas que están contribuyendo al proyecto son necesarios, pero no suficientes para demostrar una comunidad adoptante.

La comunidad Eclipse considera la ausencia de una o más de estas comunidades como prueba de que el proyecto no es lo suficientemente abierto, transparente y acogedor, y/o que se ha hecho hincapié en las herramientas a expensas de los frameworks extensibles o viceversa.

5.3.1.4 Evolución

Es un objetivo explícito del proceso de desarrollo para ser lo más claro y conciso posible con el fin de ayudar a los equipos del proyecto a navegar las complejidades,

evitar las trampas y tener éxito lo más rápido posible. Además de proporcionar la mayor libertad y autonomía para los proyectos como sea posible al tiempo que garantiza las cualidades colectivas que benefician a toda la comunidad Eclipse.

Los frameworks, herramientas, proyectos, procesos, la comunidad, e incluso la definición de calidad sigue, y seguirá, evolucionando. La creación de reglas o procedimientos que obligan a una estática instantánea de cualquiera de ellos es perjudicial para la salud, el crecimiento, y el impacto del ecosistema Eclipse.

5.3.1.5 Requerimientos

Los criterios adicionales establecidos por la Organización de Gestión Eclipse (EMO, Eclipse Management Organization) contiene requerimientos, recomendaciones y sugerencias.

Requerido: Se requieren ciertas responsabilidades y buen comportamiento de los participantes en proyectos open source Eclipse. Los proyectos que no lleven a cabo los comportamientos requeridos por la EMO, serán terminados. De acuerdo con los Principios Rectores, el número de requisitos debe mantenerse a un mínimo absoluto.

Directriz: Se recomiendan otras responsabilidades y comportamientos para las mejores prácticas. Colectivamente, hemos aprendido que los proyectos tienen más probabilidades de tener éxito si los miembros y líderes de equipo siguen estas recomendaciones. Por lo tanto, los proyectos que no las sigan, no serán penalizados por este proceso.

5.3.2 Estructura y Organización del Proyecto

Los proyectos son desarrolladores, código, sitios de descarga, líderes de proyecto, etc., pero también son los medios por los cuales el trabajo de código abierto se organiza cuando se presenta a las comunidades de desarrolladores, los adoptantes y los usuarios. Los proyectos proporcionan una estructura que ayuda a los desarrolladores exponer su trabajo a una amplia audiencia de consumidores.

Los proyectos Eclipse están organizados jerárquicamente. Un tipo especial de Proyecto, Proyectos de nivel superior, localizados en la parte superior de la jerarquía. Cada proyecto de nivel superior contiene uno o más proyectos. Cada proyecto puede a su vez contener cero o más proyectos. Un proyecto que tiene uno o más proyectos se dice que es el "Proyecto padre", siendo estos proyectos hijos denominados Sub-Proyectos o componentes, pero la elección del nombre común no cambian las características del proyecto.

Los proyectos son la entidad unitaria para:

- Committers
- Código y Lanzamientos (Releases)
- Registros IP (Propiedad intelectual)
- Sensibilización de la Comunidad

5.3.2.1 Committers

Cada proyecto tiene exactamente un conjunto de committers. Sin embargo cada conjunto de committers de un proyecto es distinta de la de cualquier otro proyecto, incluyendo sub-proyectos o Proyectos de padres. Todos Committers de un proyecto tienen los mismos derechos y responsabilidades dentro del proyecto. La partición de la responsabilidad se gestiona mediante las convenciones sociales.

Un proyecto puede, por ejemplo, dividirse en particiones lógicas de funcionalidad, es convención social que impide a los Committers de una partición lógica de hacer el trabajo apropiado en otro. Si es necesario el manejo más preciso de las responsabilidades del commiter, un proyecto debe considerar dividirse en dos o más subproyectos.

Los Committers de un proyecto tienen el derecho exclusivo de elegir nuevos Committers para su proyecto.

No hay roll-up de Committers: el conjunto de Committers en un proyecto es exactamente el mismo conjunto de personas que han sido elegidas de forma explícita en ese papel para el proyecto (es decir, ser un committer en un sub-proyecto no le da ningún derecho automático sobre el "Proyecto padre").

En la práctica, cada proyecto tiene un solo grupo Unix de sus Committers que proporciona acceso de escritura a los recursos del Proyecto. En la figura 14, vemos que un proyecto, además de los diversos recursos y Committers que tiene, también puede tener cero o más sub-proyectos. Cada uno de estos sub-proyectos tiene su propio conjunto de Committers y recursos.

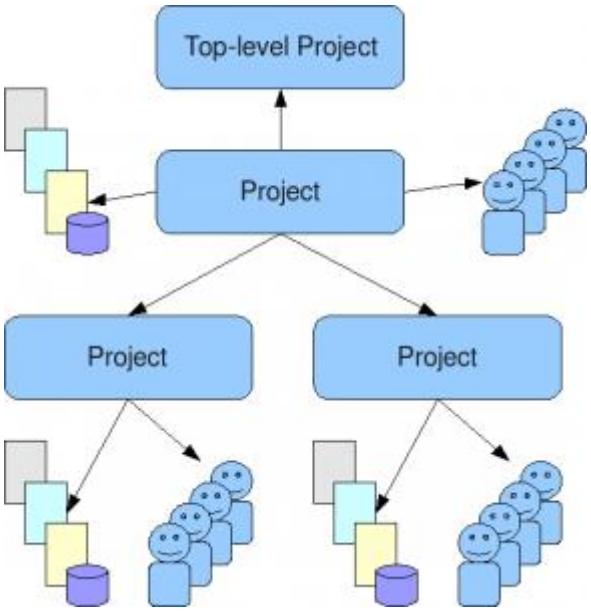

Figura 16. Niveles jerárquicos de un proyecto

Todos Committers deben estar cubiertos por un Acuerdo Committer Miembro (MCA) o un formulario de consentimiento Committer Empleador Eclipse (ECECF).

5.3.2.2 Código y Lanzamientos (Releases)

Cada proyecto posee y mantiene una colección de recursos.

Los recursos pueden incluir el código fuente, un sitio web del proyecto, el espacio en el servidor de descargas, el acceso a la construcción de los recursos y otros servicios prestados por la infraestructura de Eclipse Foundation.

Un proyecto no es estrictamente necesario para hacer uso de todos los recursos puestos a su disposición, un proyecto puede, por ejemplo, optar por no mantener un repositorio de código fuente. Tal proyecto podría funcionar como una unidad organizativa o contenedor, de varios sub-proyectos. Del mismo modo, un proyecto puede optar por proporcionar un sitio consolidado, construir y/o en el sitio de descarga de los sub-proyectos.

Cada proyecto tiene un único componente Bugzilla por sus errores.

Cualquier proyecto en la fase madura puede hacer un lanzamiento. Un proyecto en la fase de incubación, con dos mentores puede hacer un pre-1.0 Release. La puesta en libertad puede incluir el código de cualquier subconjunto de los descendientes del Proyecto.

5.3.2.3 Registros IP (Propiedad Intelectual)

Un proyecto en cualquier nivel puede recibir autorización IP para las contribuciones y las bibliotecas de terceros. La aprobación IP a menudo incluyen la misma homologación de todos los proyectos descendientes.

5.3.2.4 Sensibilización de la Comunidad

Los proyectos pueden o bien tienen sus propias comunicaciones (web, listas de correo, foros, grupos de noticias, etc), o pueden ser parte de las comunicaciones de un proyecto principal. En cualquiera de los casos, se requiere el Proyecto de mantener un canal de comunicación abierto y público con la inclusión de la comunidad Eclipse, pero no limitado a, los planes del proyecto, horarios, discusiones de diseño, y así sucesivamente.

Cualquier proyecto en la fase de incubación debe identificar correctamente su sitio web y comunicados. Un proyecto con al menos un proyecto descendiente en fase de incubación debe anotar correctamente su propio sitio web con el fin de notificar a la comunidad Eclipse que existen proyectos en incubación en su jerarquía. Cualquier publicación que contiene el código de un proyecto de la Fase de incubación deben etiquetarse correctamente, es decir, la fase de incubación es viral y se expande para cubrir todos los lanzamientos en el que está incluido.

5.3.2.5 Alcance del Proyecto

Cada proyecto de nivel superior tiene una Carta que describe el propósito, alcance, y las reglas de operación para el proyecto de nivel superior. La Carta debe hacer referencia a, y describir las mejoras a las disposiciones de este proceso de desarrollo. La

Board of director de la Fundación Eclipse aprueba la Carta de cada proyecto de nivel superior.

Sub-Proyectos no tienen Cartas separadas; Sub-proyectos operan bajo la Carta de los padres del proyecto de nivel superior. Las iniciativas o código que se encuentren fuera del alcance del proyecto, puede ser causa terminación del proyecto.

El alcance de un Sub-Proyecto se define en la propuesta inicial del proyecto tal como fue revisado y aprobado por el Comité de Gestión del Proyecto (PMC) de la matriz superior de Proyecto del Proyecto y de la EMO. El alcance de un proyecto debe ser un subconjunto del alcance de su padre.

5.3.2.6 Líderes en Eclipse

Hay dos tipos diferentes de liderazgo de proyectos de Eclipse: El Comité de Gestión del Proyecto (PMC) y el jefe de proyecto. Ambas formas de liderazgo deben:

- Asegurar que su proyecto está funcionando eficazmente, guiando la dirección general y la eliminación de obstáculos, resolver problemas y resolución de conflictos;
- Operar con las reglas de código abierto de participación: la meritocracia, la transparencia y la participación abierta, y
- Asegurar que el proyecto y sub-proyectos (si los hay) se ajustan a la política de propiedad intelectual Eclipse Fundación y Procedimientos.

El liderazgo de un proyecto se compone de Líder de Proyecto del Proyecto (s), la dirección del proyecto principal (si existe) y los jedes de PMC y miembros del PMC del Proyecto de nivel superior.

5.3.2.6.1 Comité de Gestión del Proyecto (PMC)

Un PMC tiene uno o más jefes PMC y cero o más miembros PMC. Juntos, el PMC proporciona supervisión y liderazgo global de los proyectos que se enmarcan en su proyecto de nivel superior. El PMC como un todo, y los jefes PMC en particular, son en última instancia responsables de asegurar que el proceso de desarrollo de Eclipse es entendido y seguido por sus proyectos. El PMC es, además, responsable del mantenimiento de los estatutos del proyecto de nivel superior.

Los jefes PMC son aprobados por la board of directors; los miembros PMC son elegidos por los jefes y miembros PMC existentes y aprobados por la Organización de Gestión Eclipse (EMO).

5.3.2.6.2 Líder del Proyecto

Los proyectos Eclipse son gestionados por uno o varios jefes de proyecto. Son responsables de asegurar que los Committers de su proyecto están siguiendo el proceso de desarrollo de Eclipse, y que el proyecto está participando en el tipo correcto de las actividades a desarrollar. El liderazgo inicial del proyecto es designado y aprobado en la revisión de la creación. Posteriormente, los jefes adicionales del proyecto deben ser elegidos por los Committers del proyecto y aprobados por el PMC

del Proyecto y la EMO. Un miembro puede ser removido solo por la votación unánime del resto de los jefes del proyecto.

En situaciones excepcionales, donde los proyectos tienen cero committers activos o proyectos con Committers disruptivas, el líder tiene la autoridad para hacer cambios (agregar, quitar) del conjunto de committers y/o jefes de proyecto.

5.3.2.7 Mentors

Las nuevas propuestas que intentan hacer un lanzamiento están obligados a tener al menos dos mentores. Solo aquellas propuestas que lanzan un código de un proyecto padre no están obligados a tener mentores. Los mentores deben ser miembros del Consejo de Arquitectura. Los mentores (incluyendo el nombre, afiliación y actuales proyectos/roles Eclipse) deben estar inscritos en la propuesta. Los mentores están obligados a controlar y asesorar el nuevo proyecto durante la fase de incubación, pero son liberados de esta obligación una vez que el proyecto se gradúa a la fase de madurez.

5.4 Ciclo de Vida de un Proyecto

Los proyectos pasan por seis fases distintas. Las transiciones de fase a fase son abiertas y transparentes a opiniones públicas.

5.4.1 Pre-propuesta

Un individuo o grupo de individuos declara su interés, y razonamiento de establecer un proyecto. La EMO ayudará a estos grupos en la elaboración de una propuesta de proyecto.

La fase de Pre-propuesta termina cuando la propuesta es publicada por la EMO y anunciada a sus miembros.

5.4.2 Propuesta

Los proponentes, en relación con el destino del PMC y la comunidad, para colaborar en público para mejorar, perfeccionar y aclarar la propuesta. Los mentores (si es necesario) para el proyecto deben ser identificados durante esta fase.

- La fase de propuesta finaliza con una revisión de creación, o el retiro.
- La propuesta puede ser retirada por sus autores.
- La EMO retire una propuesta que ha estado inactivo durante más de seis meses.

5.4.3 Incubación

Después de que se ha creado el proyecto, el propósito de la fase de incubación es establecer un proyecto de código abierto en pleno funcionamiento. En este contexto, la incubación se trata de desarrollar el proceso, la comunidad, y la tecnología. La incubación es una fase más que un lugar: nuevos proyectos pueden ser incubados bajo cualquier proyecto existente.

- La fase de incubación puede continuar con una revisión continua o con una revisión de lanzamiento.
- Proyectos de nivel superior no se pueden incubar y sólo pueden ser creados a partir de uno o más proyectos en fase de madurez existentes o de nivel superior.
- La fase de incubación termina con una revisión de la graduación o una revisión de terminación.
- Los proyectos designados en la Incubadora pueden permanecer perpetuamente en la fase de incubación; las revisiones son obligatorias.

Un proyecto podrá designar a un Sub-Proyecto como una "Incubadora". La incubadora es un excelente lugar para innovar, probar nuevas ideas, crear funcionalidad que algún día podría ser movido a otro proyecto, y desarrollar nuevos committers.

Incubadora de Proyectos no tienen liberaciones; no requieren revisiones anuales de continuación y no son parte del tren de lanzamiento anual.

El alcance de una Incubadora de Proyectos debe entrar en el ámbito de su proyecto padre. El grupo committer del Proyecto Incubadora debe coincidir con la del proyecto padre (al menos un committer del proyecto padre debe ser un committer de la incubadora).

5.4.4 Proyecto Maduro

El equipo del proyecto ha demostrado que es un proyecto open source con un proceso abierto y transparente, una comunidad que participa activamente y en crecimiento, y la tecnología Eclipse es de calidad. El proyecto es ahora un miembro maduro de la comunidad Eclipse. Los lanzamiento principales siguen pasando por revisión de versiones.

- Proyectos de fase maduros tienen lanzamientos a través de revisiones de la versión.
- Un proyecto maduro puede ser promovido a un proyecto de nivel superior a través de una revisión de la promoción.
- Un proyecto maduro que no participe en un lanzamiento en el año determinado podrá seguir a través de una revisión continua.
- Los proyectos de fase maduros inactivos pueden ser archivados a través de una revisión de terminación.

5.4.5 Proyecto de Nivel Superior

Los proyectos que han demostrado las características de un proyecto de nivel superior (por ejemplo, un liderazgo consistente en un área técnica y la contratación de una comunidad de desarrolladores en general) pueden ser promovidos a la categoría de nivel superior del proyecto. Esta promoción se produce a través de una revisión de la Promoción. Tras la finalización con éxito de la revisión de promoción, la EMO puede

recomendar que el proyecto sea promovido a la Board of directors y pedir que se revise y aprobé su carta.

5.4.6 Proyecto Archivado

Los proyectos que se convierten en inactivos, ya sea a través de la disminución de los recursos, o por llegar a su conclusión natural, se archivan. Los proyectos pueden llegar a su conclusión natural en un número de maneras: por ejemplo, un proyecto podría llegar a ser tan popular que se absorbe en uno de los otros frameworks principales. Los proyectos se trasladaron al estado archivados a través de una revisión de terminación.

Si hay suficiente interés de la comunidad en la reactivación de un proyecto archivado, el proyecto se pondrá en marcha de nuevo con Revisión Creación. Como debe haber buenas razones para haber movido un Proyecto a los Archivos, la Revista de creación proporciona un nivel suficientemente alto para probar que esas razones ya no son válidas.

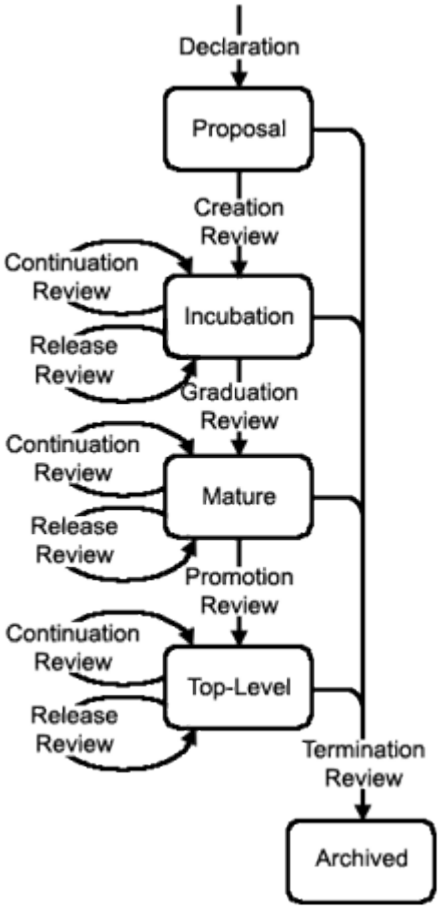

Figura 17. Ciclo de vida de un proyecto

5.4.7 Revisión de Creación

El propósito de la revisión de creación es para evaluar la respuesta de la comunidad y la pertenencia a la propuesta, para verificar que los recursos necesarios están disponibles para el proyecto para lograr su plan, y para servir como una elección de

committer para Committers iniciales del proyecto. La Eclipse Foundation se esfuerza por no ser un repositorio de "vertederos de código" y por lo tanto, los proyectos deben estar suficientemente dotados de personal para el progreso hacia adelante.

Los documentos de la revisión de creación debe de incluir biografías cortas de los commiters propuestos inicialmente. Estas biografías deben discutir su relación con la historia del código de entrada y/o de su participación en el área tecnológica cubiertas por la propuesta. El objetivo es ayudar a mantener un legado de contribuyentes conectados al nuevo proyecto y explicar la conexión a la membresía Eclipse actual y futura, así como justificar la participación de los Committers iniciales en la meritocracia.

5.4.8 Revisión de Graduación

El propósito de la revisión de la graduación es la confirmación de que el proyecto es y tiene:

- Una base de código bien trabajado y demostrable de calidad suficientemente alta.
- Comunidades activas y suficientemente diversas adecuadas para el tamaño de la base de código de graduación: adoptantes, desarrolladores y usuarios.
- Pleno funcionamiento en su apertura siguiendo los principios y propósitos de Eclipse.
- Un crédito para Eclipse y estar funcionando bien dentro de la comunidad Eclipse.

La revisión de la graduación es sobre el cambio de fase incubación a la fase madura. Si el proyecto y/o algunos de su código se está trasladando simultáneamente a otro proyecto, la revisión de graduación se combinará con una revisión de reestructuración.

5.4.9 Revisión de Lanzamiento

Los efectos de una revisión de lanzamiento son: resumir los logros de la liberación, para comprobar que la política de propiedad intelectual se ha seguido y se han recibido todas las aprobaciones, para destacar alguna cualidad restante y/o problemas de arquitectura, y para verificar que el proyecto continuara operando de acuerdo con los principios y propósitos de Eclipse.

5.4.10 Revisión de Promoción

El propósito de una revisión de promoción para determinar si el proyecto ha demostrado las características de un proyecto de nivel superior, debe tener por ejemplo, un liderazgo consistente en un área técnica y la contratación de una comunidad de desarrolladores en general. El proyecto tiene que ser un proyecto Eclipse maduro que funcione bien, por lo que la evidencia contraria será un punto negativo para la promoción. Los proyectos de nivel superior, tanto a través de su existencia y de los miembros del Consejo, tienen gran influencia sobre la dirección y el

funcionamiento de Eclipse, por lo que corresponde a los miembros conceder la condición de nivel superior sólo por mérito: por el servicio demostrado a lo largo del ecosistema Eclipse.

5.4.11 Revisión de Continuación

El propósito de una revisión de continuación, es para verificar que una propuesta o proyecto sigue siendo un esfuerzo viable y un crédito para Eclipse. Se espera que el equipo del proyecto explique los avances técnicos recientes y demuestre suficientes adoptantes, desarrolladores y usuarios de soporte para el proyecto. El objetivo de la revisión de continuación es para evitar que los proyectos inactivos que buscan prometer, pero que en realidad nunca hacen entrega de frameworks extensibles y herramientas ejemplares para el ecosistema.

5.4.12 Revisión de Terminación

El propósito de una revisión de terminación es proporcionar una oportunidad final para los committers y/o miembro de Eclipse, para discutir la oferta de retirada de una propuesta o el archivo de un proyecto. El resultado deseado es encontrar pruebas suficientes para renovar el interés y los recursos para mantener el proyecto o la propuesta activa. Si no se encuentran se termina.

5.4.13 Revisión de reestructuración

El propósito de una revisión de reestructuración, es notificar a la comunidad de su intención de realizar cambios significativos en uno o más proyectos. Estos cambios incluye:

- Movimiento de trozos importantes de la funcionalidad de un proyecto a otro;
- La modificación de la estructura del proyecto, por ejemplo, la combinación de múltiples proyectos en un solo proyecto, o descomponer un solo proyecto en varios proyectos, y/o
- Cambio del alcance del proyecto.

Una revisión reestructuración puede incluir el movimiento de trozos significativos de código. Un movimiento se considera significativo si se tiene un impacto en la comunidad. Esto puede incluir proyectos enteros, lotes y características, pero es probable que excluye pequeños fragmentos, fragmentos de código y archivos individuales.

5.5 Tamaño de Eclipse

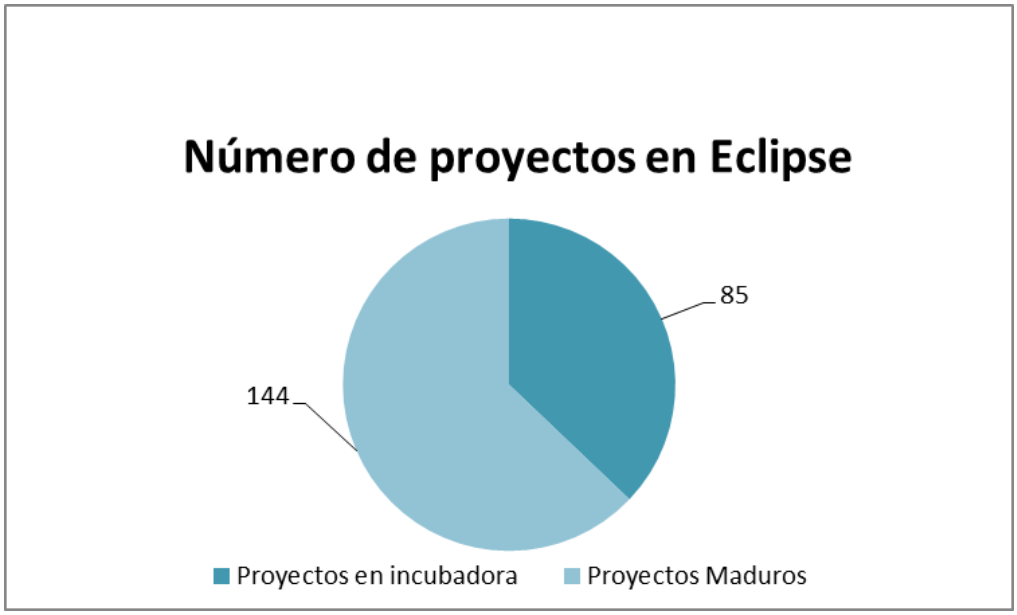

Figura 18. Número de proyectos en Eclipse

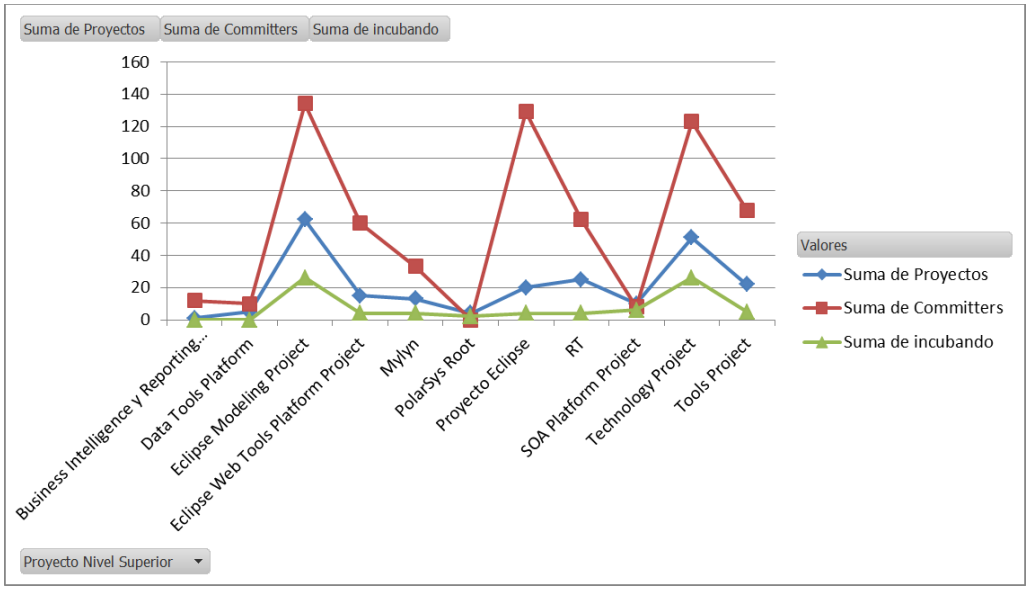

Figura 19. Proyectos graduados con relación a committers y tiempo de incubación en meses

6 Análisis

	Incubador Apache	Incubador Eclipse
Definición del proceso de incubación	1. Fue establecido en 2002. 2. Este es descrito, con etapas y actividades específicas. 3. La política de incubación, principios y la filosofía son públicos (http://incubator.apache.org/incubation/Incubation_Policy.html).	1. Fue establecido en el 2003. 2. Es descrito en un documento legal en www.eclipse.org/org/documents y en la Eclipse Web site. 3. La política de incubación y principios son públicos (www.eclipse.org/projects/dev_process).
Estructura organizacional	1. Los roles y responsabilidades son claramente definidos. 2. La estructura está centralizada: La administración de proyecto Committeer es responsable de la administración del incubador.	1. Los roles y responsabilidades son claramente definidos 2. La estructura está centralizada: la tecnología de alto nivel del proyecto PMC es responsable de la incubación de los proyectos.
Medida de mitigación de riesgos	1. La propuesta de creación y revisión son archivados y están accesibles. 2. . Los proyectos puedes graduarse con subproyectos o proyectos de alto nivel. 3. Los proyectos siguen un enfoque iterativo durante la fase de revisión para lograr una primera versión estable.	4. Al principio del proceso durante la creación de la revisión, la organización de la dirección en Eclipse evalúa el tamaño de la comunidad. En la fase de propuesta, el PMC define maneras de participar como un alto nivel de proyectos o subproyectos. 5. Los proyectos siguen un enfoque iterativo durante la incubación/validación, para lograr una primera versión estable.
La información del proceso de publicación	1. Las propuestas de proyectos e informes de la junta se publican en el Incubator wiki (http://wiki.apache.org/incubator). 2. La autorización de la propiedad intelectual esta en http://incubator.apache.org/ip-clearance/index.html. 3. La información de estado de los proyectos en incubación, graduados de	1. La propuesta de creación y revisión son archivados y están accesibles en la propuesta de Eclipse Web site (www.eclipse.org/proposals). 2. Las opiniones son archivadas y accesibles en (www.eclipse.org/projects/previous-release-reviews.php). 3. Los informes de las revisiones periódicas de los proyectos y de las minutas se publican en (www.eclipse.org/technology/pmc-minutes.php).

	incubación y retiro de incubación están en http://incubator.apache.org/projects/index.html.	
Infraestructura para apoyar las actividades de desarrollo de los proyectos	1. Incubador de sitios web y wikis . 2. Repositorios de código 3. Descarga de sitios y sistemas de espejos de distribución. 4. Entorno de gestión de correo. 5. Edición y seguimiento de fallos.	1. Sitio web del proyecto 2. Repositorio de código 3. Descarga de sitios y sistemas de distribución de duplicación. 4. Entorno de gestión de correo Edición y seguimiento de fallos

7 Conclusiones

Concluimos que a partir de la información obtenida sobre los procesos de los ecosistemas de código abierto en las comunidades Apache y Eclipse, ambos enfoques se diferencian entre proyectos de nivel superior y subproyectos. Esto es muy relevante, ya que reconoce que muchas de las iniciativas de proyectos de código abierto nacen dentro de una comunidad ya formada. Por otra parte, los proyectos exitosos a menudo promueven subproyectos que eventualmente adquieren el status de los proyectos de nivel superior.

Ambas incubadoras requieren que los candidatos se han centrados en un conjunto de objetivos que deben declararse formalmente en una propuesta. Esto ayuda a crear comunidades saludables que apuntan a resolver problemas específicos, promover la eficacia.

También hacen hincapié en la importancia de la comunidad para la salud de los proyectos utilizando el número de committers como un criterio para su aprobación. Esto tiene como resultado proyectos con larga vida que seguirán creando valor, incluso si el conjunto inicial de committers los abandona.

Además siguen un enfoque iterativo durante la fase de incubación, reduciendo el riesgo asociado al entregar una primera versión estable.

A pesar de sus similitudes, Apache y Eclipse abordan el proceso de incubación de formas ligeramente diferentes: Apache ha creado una estructura organizativa específica. La incubadora, así es un PMC que se dedica a garantizar el éxito del proyecto candidato (http://incubator.apache.org/official/resolution.html). Eclipse prefiere incubar nuevos proyectos bajo la estructura de un proyecto de nivel superior existente. El PMC de nivel superior debe hacer una importante inversión de tiempo y energía.

También concluimos que para formar un ecosistema de código abierto auto sostenible como la fundación Apache y Eclipse. Es necesario y fundamental observar las tendencias en los modelos de negocio. Y definir el propio modelo que se ajuste a los intereses de los actores internos y externos, que conformaran nuestro ecosistema de código abierto auto sostenible. Una vez identificados, se establece una estructura jerárquica organizacional que este fundamentada en los principios básicos de la ingeniería de software de código abierto, con la finalidad de crear, divulgar y distribuir conocimiento de código. Pero con la importancia de crear los mecanismos de financiación, distribución, comunicación, publicación, oferta a usuarios y captación de proyectos que alimenten y retroalimenten, nuestro ecosistema.

Bibliografía

1. Juan C. Dueñas, Hugo A. Parada G., Félix Cuadrado, Manuel Santillán, and José L. Ruiz, Universidad Politécnica de Madrid. "Apache and Eclipse: Comparing Open Source Project Incubators". 2007 The IEEE Computer Society, Páginas 90-98.
2. Konstantinos Manikas, Klaus Marius Hansen. "Software ecosystems: A systematic literature review". Journal of Systems and Software. Volumen 86, Edición 5, Mayo 2013, Páginas 1294–1306. ISSN: 0164-1212. Última vez revisado Mayo 2013. http://www.sciencedirect.com/science/article/pii/S016412121200338X
3. Rodrigo Santos, Cláudia Werner. "Software ecosystems: Trends and impacts on software engineering". 2012 Brazilian Symposium on Software Engineering. Última vez revisado Mayo 2013. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6337876
4. Rodrigo Pereira dos Santos, Cláudia Maria Lima Werner. "ReuseECOS: An Approach to Support Global Software Development through Software Ecosystems". 2012 IEEE Seventh International Conference on Global Software Engineering Workshops. Última vez revisado Mayo 2013. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6337321
5. Slinger Jansen,Michael Cusumano. "Defining Software Ecosystems: A Survey of Software Platforms and Business Network". 2012 International Workshop on Software Ecosystems IWSECO. Última vez revisado Mayo 2013. http://ceur-ws.org/Vol-879/paper4.pdf
6. Sami Hyrynsalmi, Tuomas Mäkilä, Antero Järvi, Arho Suominen, Marko Seppänen, Timo Knuutila. "App Store, Marketplace, Play! An Analysis of Multi-Homing in Mobile Software Ecosystems". 2012 International Workshop on Software Ecosystems IWSECO. Última vez revisado Mayo 2013. http://ceur-ws.org/Vol-879/paper5.pdf
7. Bosch, J., "From software product lines to software ecosystems". Proceedings of the 13th International Software Product Line Conference. Carnegie Mellon University, Pittsburgh, PA, USA, pp. 111–119. 2009
8. Lungu, M., Lanza, M., Girrba, T., Robbes, R.: The small project observatory: Visualizing software ecosystems. Science of Computer Programming 75 (2010) 264-275.
9. Brooks, Frederick P., J.: The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley (1975).
10. DeMarco, T., Lister, T.: Peopleware: productive projects and teams. Dorset House Publishing (1987).
11. J. Lerner and J. Tirole, "Economic Perspectives on Open Source," in Perspectives on free and open source software, vol. Volume 15, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani, Eds. MIT Press, 2005, pp. 47-78.
12. T. Jaeger, O. Koglin, T. Kreutzer, A. Metzger, and C. Schulz, "Die GPL kommentiert und erklärt," in Die GPL kommentiert und erklärt, Institutfr Rechtsfragen der Freien und Open Source Software, O'Reilly, 2005, pp. 1-24.
13. K. Popp and R. Meyer, Profit from Software Ecosystems: Business Models, Ecosystems and Partnerships in the Software Industry [Paperback]. Books on Demand, 2010, p. 242.

14. J. Lindman and R. Rajala, “How Open Source Has Changed the Software Industry: Perspectives from Open Source Entrepreneurs,” Technology Innovation Management Review, no. January, pp. 5-11, 2012.
15. K. J. Bekkelund, “Succeeding with freemium,” NTNU, 2010.
16. N. Pujol, “Freemium: attributes of an emerging business model,” Organization, no. December, 2010
17. D. Ascher, “Is Open Source Right for You?: A fictional case study of open source in a commercial software shop,” Queue, vol. 2, no. 3, pp. 32-38, 2004.
18. http://en.wikipedia.org/wiki/Business_models_for_open-source_software
19. http://incubator.apache.org/incubation/Roles_and_Responsibilities.html#Roles+and+Responsibilities
20. http://incubator.apache.org/guides/graduation.html
21. http://www.apache.org/foundation/marks/responsibility
22. http://www.apache.org/foundation/records/
23. http://www.apache.org/foundation/
24. http://incubator.apache.org/projects/index.html
25. http://www.apache.org/licenses/
26. http://incubator.apache.org/projects/
27. http://incubator.apache.org/guides/proposal.html
28. http://www.eclipse.org/projects/dev_process/development_process.php
29. http://wiki.eclipse.org/Development_Resources/HOWTO/Nominating_and_Electing_a_New_Committer
30. http://www.eclipse.org/projects/dev_process/development_process_2011.php#1_Purpose
31. http://projects.apache.org/indexes/category.html